ARI Research Note 2002-06

Intelligent Dialog Tutor and Conversational Agents -- Continuation

Michelle Sams and William R. Murray
Teknowledge Corporation

William H. DeSmedt and Donald Loritz
Amber Consortium

Research and Advanced Concepts Office

February 2002

U.S. Army Research Institute
for the Behavioral and Social Sciences

U.S. Army Research Institute
for the Behavioral and Social Sciences

A Directorate of the U.S. Total Army Personnel Command

ZITA M. SIMUTIS
Acting Director

Research accomplished under contract
for the Department of the Army

Teknowledge Corporation

Technical review by

Jonathan Kaplan

NOTICES

REPORT DOCUMENTATION PAGE

1. REPORT DATE (dd-mm-yy)	2. REPORT TYPE	3. DATES COVERED (from. . . to)
February 2002	Final	February 1999-January 2002

4. TITLE AND SUBTITLE	5a. CONTRACT OR GRANT NUMBER
Intelligent Dialog Tutor and Conversational Agents	DASW01-99-C-0021
	5b. PROGRAM ELEMENT NUMBER 2O665502
6. AUTHOR(S): Michelle Sams and William R. Murray (Teknowledge Corp.), William H. DeSmedt and Donald Loritz (Amber Consortium)	5c. PROJECT NUMBER M770
	5d. TASK NUMBER
	5e. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Teknowledge Corporation 4350 N. Fairfax Drive, Suite 420 Arlington, VA 22203	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Institute for the Behavioral and Social Sciences Attn: TAPC-ARI-BR 5001 Eisenhower Avenue Alexandria, VA 22333-5600	10. MONITOR ACRONYM ARI
	11. MONITOR REPORT NUMBER Research Note 2002-06

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES.

14. ABSTRACT *(Maximum 200 words)*: The report describes research done to develop a prototype of an easy to use authoring system for an intelligent, computer-based trainer. It focuses on simulated agents who are capable of mixed-initiative dialog with the user through a natural language interface. Trainees converse as they would in normal English discourse asking quests, giving answers, and making comments. The simulated agent generates appropriate responses within any defined domain. The resulting tutor provides individualized instruction based on trained performance. The system is fully authorable and conversational agents are easily created for new scenarios. Technologies used include: natural language processing, semantic analysis, discourse management, agent control architectures, and distributed object-oriented software.

15. SUBJECT TERMS

Intelligent tutor, authoring system, computer-based training, natural language processing, intelligent agents.

SECURITY CLASSIFICATION OF			19. LIMITATION OF ABSTRACT	20. NUMBER OF PAGES	21. RESPONSIBLE PERSON (Name and Telephone Number)
16. REPORT	17. ABSTRACT	18. THIS PAGE			
Unclassified	Unclassified	Unclassified	Unlimited	53	Jonathan Kaplan (703) 617-8828

Intelligent Dialog Tutor and Conversational Agents

CONTENTS

INTRODUCTION

The Army is undergoing transformation of its force and of its approach to meet their training needs. Training will be organized around real-time applications of information and information technology on the battlefield, in the units, and in the classroom to maintain a continuous edge in projecting and employing combat power. The proposed effort supports the Army's Objective Force for computer-based training on stand-alone platforms or in distributed simulated environments to deliver training when and where needed.

Most current computer-based training is electronic page turning, with some hyperlinks and multiple-choice exercises. While this electronic delivery provides self-paced learning, it does not adapt to the trainee and is not truly interactive. One-on-one human tutorial interactions have been found to be the most instructionally effective (Bloom, 1984). This is largely due to the human tutor's ability to reason about the domain, to construct a model of the individual learner, and to adapt the instructional strategy when in a one-to-one situation. However, another major contributor to effectiveness, that is often overlooked, lies in the communication interface -- the natural dialog. The tutor can ask questions and understand the learner's responses, and the learner can ask clarification questions and receive appropriate explanations. It is through this negotiation of meaning and understanding that learning occurs.

Enabling a computer to converse as a human has been a research goal for many years. A classic test proposed by Allan Turing is one in which a human cannot determine whether they are communicating with another human or a machine (Platt, 1995). There is currently no system that can converse as smoothly, broadly, and in depth as a human. However, there have been significant technological strides and successful applications within specified limitations and domains.

The goal of this project is to develop simulated agents who are capable of mixed-initiative dialog with the user through a natural language interface. Trainees converse as they would in normal English discourse -- asking questions, giving answers, and making comments or requests. The simulated agent responds appropriately to these inputs based on what the agent "knows" and its goals.

The Intelligent Dialog Tutor will provide trainees with individualized instruction through natural language dialog with a simulated tutor agent. The tutor presents instruction and asks questions of the trainee. The trainee provides a freely formed response. The Tutor then evaluates the trainee's responses and gives specific feedback. Trainees can ask questions about the material at any time. After the lesson, the trainee can practice the newly acquired skills in scenarios populated with conversational agents.

Our dialog agents will provide intelligent and realistic training environments that are easy to author by non-programmers. This meets the Army's need for just-in-time training for the rapidly changing missions and new battlefield technology.

FUNCTIONAL OVERVIEW

First we will provide a brief, non-technical overview of the functionality of the Tutor, conversational agents, and authoring procedure. A more detailed specification of the technical approach follows.

Tutor Interaction

The Tutor presents an overview of the instructional material. Trainees can click on "Learning Object" to view any associated multimedia. The Tutor then asks the trainee a question about the instructional material. A response box is provided for the trainee's freely typed input. After the trainee responds to the question, it is evaluated for accuracy and completeness. The Tutor then provides the appropriate response. A correct evaluation will trigger the Tutor to give affirmative feedback. For a correct but incomplete response, the Tutor will indicate which parts of the answer were correct and which items were not correct or missing. An incorrect evaluation will trigger the Tutor to provide feedback about the expected response.

The Tutor evaluates the trainee's response by comparing it to an exemplar answer that was entered and stored during the authoring process. Answers do not have to match the exemplar word for word, as they are evaluated for semantic equivalence using the dialog system capabilities. They just have to 'mean the same thing' to be evaluated as correct.

What guides the overall instructional process is the Tutor's Curriculum Information Network (CIN) and the currently selected instructional strategy. The CIN represents the topics and skills to teach and relationships among them (e.g., general—specific, prerequisite-of—has-prerequisite). The instructional strategy interprets the subject matter representation in the CIN to select topics and actions. Currently, a simple top-down layered instructional strategy is functional in the Tutor, although the architecture has the flexibility of handling multiple strategies.

The correctness or incorrectness of the user's answers to the questions is used to update a user model. The user model provides a probability distribution over different possible skill levels for the student. As more correct answers accumulate, this distribution favors an interpretation where the student has a high-level of skill. As more incorrect answers accumulate, the distribution will shift to lower levels of skill. Currently three skill levels (novice, intermediate, expert) are used. The techniques of updating are based on Bayesian probability propagation that need not concern us here (for details see Pearl, 1997).

The user model is used to gauge the success of the instructional strategy. If the instructional strategy fails, then repair instructional strategies are invoked. Currently there is only one functional repair strategy: to review all topics that were not adequately mastered. Again, the instructional plan interpreter's architecture is flexible enough to support multiple attempts at re-training, so the tutor could try first one approach then another, but only the top-down repair strategy has been implemented at this time.

The trainee has the option to ask the Tutor a question at any time while progressing through the lesson material. We call this interaction, side-bar conversations. The Tutor evaluates the trainee's query and responds appropriately. It generates the response based on facts contained in its own tutorial knowledge base or in the common knowledge base. It answers the question and then resumes its tutorial strategy by repeating the last unanswered tutorial question.

Conversational Agent Interaction

Conversational agents (CA) are agents with different world-views (belief systems). An author creates world-views by entering facts to the character's own knowledge base. Conversational agents also have access to the common knowledge base.

Conversational agents have basic discourse goals that prompt them to ask questions of the user. The basic dialog strategy is to pose a question to the user when the user has stopped asking questions or makes a statement that requires no response. Although Conversational agents can talk about a subject matter domain they do not have tutorial strategies. Users can ask questions of conversational agents at any time, and the conversational agent will generate a response based on facts contained in its own belief system (knowledge base) or common sense knowledge base.

Authoring Process

Users can create new tutors or conversational agents, or edit existing ones. The author selects the identity elements: agent name and the three personality attributes (gender, helpfulness, and manner). Selection of helpfulness and politeness levels affects the dialog characteristics of the agents. Toward the casual-rude end of the spectrum, the agent will assume it is on a first-name basis with its collocutor, or even start insulting them.

The next authoring screen allows facts to be added to the agent's own belief set (knowledge base). A fact is entered as a complete sentence. Facts which describe the character itself are given as directions: You believe x, You are y, You like z.

After each sentence is entered as a fact, it is proofed by the dialog system. The proofing process checks to make sure that the dialog system 'understands' the words and the sentence. It then transforms the sentence to its own internal format. If the dialog system cannot parse the sentences or the questions, or if unknown terms are used, then the dialog system will italicize the unknown text. Individual unknown terms need to be entered into the dialog system. Parts of speech are identified as well as its position in the ontology (a concept tree). This authoring process is described later.

If the author wants all Conversational agents and Tutor agents to have access to the same knowledge (facts), then these facts are added to the Common Knowledge base.

SYSTEM AND COMPONENTS OVERVIEW

Here we will give a brief non-technical overview of the components in our Dialog System. Full technical descriptions follow. Our dialog system provides the capability for the agents to recognize, understand, and generate language -- "how to converse". Our knowledge representation system provides the agents with general concepts and specific facts -- "what to talk about". Tutorial strategies provide our agents with goals and planning capability -- "how and when to say it".

Dialog System

The Dialog System enables the simulated agents to 'understand' freely formed user input and to generate appropriate responses. When the user poses a question or makes a statement, the input is first parsed to identify what the words are and how they are related to each other (e.g., who did what to whom). The

parser relies on several sub-components to do this job, including a morphological analyzer and a lexicon (similar to a dictionary).

The results of this syntactic analysis are enhanced with contextual information from the discourse management technology and mapped into a semantic case-frame to resolve discourse phenomena (e.g., pronouns referring to previously mentioned people and objects). After the semantic and discourse analysis, the input is reduced to one or more propositions (or "factoids") expressed in a language-independent formalism. The factoid is now passed to the knowledge manager. The knowledge manager's inference engine attempts to match the incoming factoid against the contents of the currently active knowledge base.

First let us consider the case where the user has asked the Dialog System a question. The dialog system formulates a candidate response by matching against the knowledge bases. The candidate response factoid is then run back through a discourse-management filter to screen out elements that are not germane to the current discussion, and to accord appropriate emphasis to those which are. Personal asides that reflect a character's personality, such as saying "You fool…", "As I already said…", or "Sir" are added at this state, too. The resulting discourse-sensitive factoid is then fed into the language generator, which performs a mapping from semantic to syntactic case frames, applies surface transforms consistent with the focus of discourse, and formulates an output string for delivery to the learner.

Now let us consider what happens when the tutor asks a question and the trainee provides a response. Any tutor question can have multiple items in a correct response. The Dialog Agent system attempts to find the best match, ignoring order, from the student's answer (set of items) to the Tutor's pre-stored (exemplar) set of items.

Semantic equivalence is used to compare trainee response items to exemplar answer items. First the items are compared for their position in the ontology to see if one is above or below another. This score is reflected as an item's generality score (e.g., less specific than exemplar). Next, the items are compared for completeness: does the student's answer have all the details of the exemplar answer? This second score is the item's comprehensiveness score. Each item is given these two scores: generality and comprehensiveness. Semantic equivalence allows two synonyms to match (e.g., HQ for headquarters) and different phrasings to be taken into account (e.g., 'John's book' is the same as 'the book of John' or 'the book that John has').

The Dialog Agent System then provides feedback on whether the expected items were recognized in the student's answer. For example, given the query "What are the results of the IPB step?" with Tutor exemplar answers "Modified Combined Obstacle Overlay" and "Commander's Critical Information Requirements". For a student answer of "An MCOO and a terrain analysis" the tutor responds:
"1 item correct: an MCOO is a result of the IPB step. 1 item was incorrect: I was looking for CCIR as the result of the IPB step."

Knowledge Representation

Each Dialog agent (Tutor and conversational agent) has their own knowledge base – facts pertaining to themselves and their particular domain. Each agent also inherits global knowledge from a central knowledge base, called Common Knowledge.

If there is a conflict between a fact in the agent's specific knowledge base and the common sense knowledge base, the agent will 'believe' the fact in its own knowledge base. These individual belief systems are useful for creating characters with different experiences and knowledge. For example, an

author can create historical characters with different beliefs. If the common sense knowledge base states that the world is round, then all characters will believe that the earth is round unless a contrary statement is entered into the character's own knowledge base.

The dialog system can make simple deductions about subject matter propositions and propositions about themselves. For example, given the following rules in Common Knowledge:

> All lawyers are rich.
> All rich people have large houses.

and the following fact in the belief set of a the specific Conversational agent, Mr. Darrow:

> You are a lawyer.

The system can infer from the common knowledge base that Darrow has a large house and is rich, even though it was not explicitly authored in that specific character's knowledge base.

> Questioner: Do you have a large house?
> Mr. Darrow: Of course I have a large house.
> Questioner: Are you rich?
> Mr. Darrow: I am rich.

Intelligent Tutoring System

The intelligent tutoring system drives the selections of topics and actions (e.g., present or assess). It models user knowledge of the topics in a user model, which is updated based on the answer analysis discussed previously. An overall instructional strategy directs the shifting focus of instruction from one topic to related topics, and the different kinds of actions invoked for each topic. An instructional plan interpreter interprets the instructional strategies for any authored CIN.

The current operational instructional strategy expects a tree-shaped representation of the material to be learned. This Curriculum Information Network (CIN) represents the target skill to be learned and its constituent sub-skills, and depicts how they are related. The Top-Down Layered Instructional strategy first traverses the nodes from the top root node to the leaves, providing an overview for each topic. Next the traversal of nodes starts at top and proceeds to the leaves again, this time providing a more detailed discussion of each topic. A third traversal asks questions for each topic to evaluate how well the material has been learned.

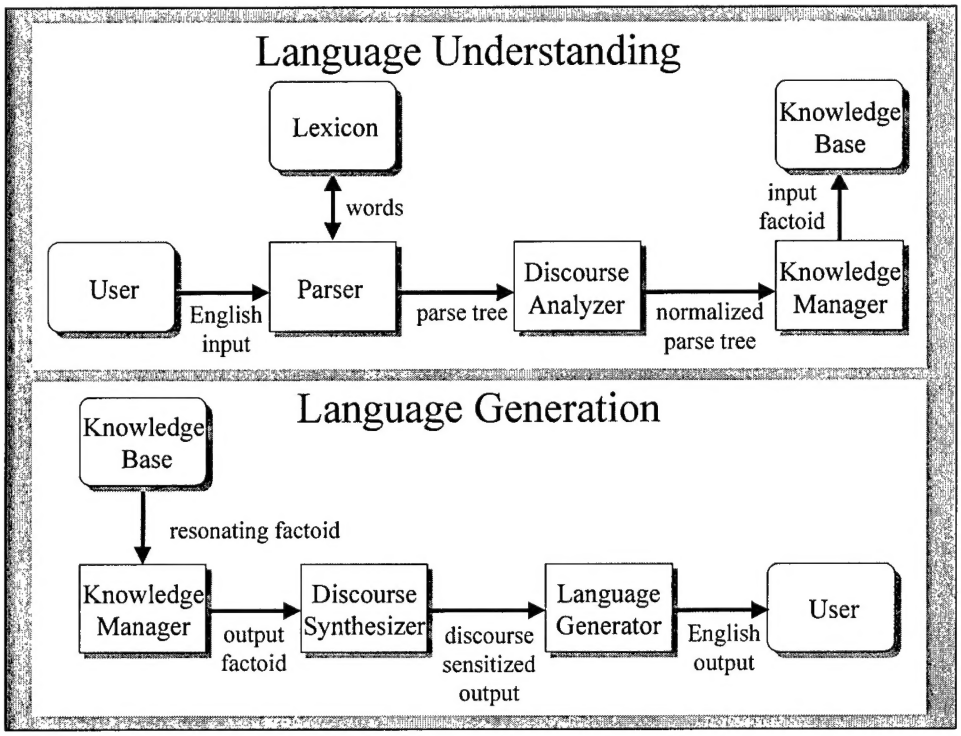
The strategy is considered to succeed if the student can answer most questions correctly (updated by Bayesian probability propagation, although other means of user modeling could also be used). It fails if the student's answers indicate the target skill was not learned adequately. If one plan fails to achieve a target skill then a repair plan is tried. Currently there is only one repair plan: an instructional strategy to traverse the CIN from the top-down and invoke repair actions only on those nodes that were not understood sufficiently (as indicated by the student model that was updated by the questions asked for each node).

Such a complex mechanism is serious overkill if no greater flexibility than top-down traversal were intended. However, our approach was to build a flexible architecture that could be extended in later efforts. The architecture is intended to support multiple instructional strategies, both for presentation and

repair that apply to any CIN. Then a developer could use the same CIN and instructional materials for many different kinds of instructional activities and strategies.

DETAILS OF THE DIALOG SYSTEM

We will briefly describe key dialog systems components and the authoring process, and then give examples of capabilities and current limitations.



Components of the Dialog System

The Parser

In order to determine *who did what to whom*, the dialog system relies on its symbolic parser. This parser is capable of handling a broad range of clausal English constructions. For example:

a. SSG Bozo gave an insult to PFC Aiken.
b. He insulted PFC Aiken.
c. PFC Aiken was insulted by SSG Bozo.
d. It was PFC Aiken who was insulted by SSG Bozo.
e. What did SSG Bozo harass PFC Aiken with?

In *(a)* the semantic relations of the sentence are canonical. SSG Bozo is the *who* (the semantic "actor" or "agent"), an *insult* is the *what* (the semantic "patient" or "theme"), and PFC Aiken is the *whom* (the abstract semantic "goal", a polymorphic semantic role often realized as either a "beneficiary" or an "experiencer"). In *(b)* there is no overt *what* (the *what* has been incorporated into the verb as in "Sgt Bozo insulted an insult to Pfc Aiken"). In *(c)* the *whom* precedes the *who* (a "passive" construction). In *(d)* a passive construction is embedded within a "cleft" sentence, *It was X*. Although in *(b)* it is the task of

the dialogue manager to identify who *He* is, in *(d)* it is the task of the parser to resolve *who* to *Pfc Aiken. (e)*, a question, illustrates that the order of the principal semantic elements can be permuted, here to *what-who-whom.* There are many more permutations of these fundamental elements and the many "oblique" elements of sentences (e.g., *when, why, how, where, if, because*). The parser must disentangle all of these before further processing is possible.

The parser recursively builds upon these clausal capabilities to perform metaclausal analysis, handling such constructs as subordinate/relative and coordinate clauses. Symbolic parsers can be divided into bottom-up and top-down types. Context-free bottom-up parsers are more efficient than context-sensitive top-down parsers if, and only if; 1) all interpretations of a sentence are of interest (e.g., *Flying airplanes can be dangerous*), and 2) there are no homonyms or polysemes in the sentence.

It is normally desirable, however, for a parser to be context-sensitive, so as to return only the most relevant interpretation (e.g., only *The flying of (or on) airplanes can be dangerous,* not *Airplanes which are flying can be dangerous*). Relatively uninflected languages like English are also rich in homonyms (e.g., *return* (v) vs. *return* (n)) and polysemes (e.g., *rich* (fatty, caloric) vs. *rich* (affluent) vs. *rich* (abundant)). Because conditions (1) and (2) are rarely encountered in English natural language processing tasks, top-down symbolic parsers are much more efficient in practice.

Top-down symbolic parsers can, in turn, be divided into two broad types: definite clause grammar (DCG) parsers and augmented transition network (ATN) parsers. Simple definite clause grammars consist of simple rules and are easy to author for small domains. Simple ATN grammars are more code-like and are harder to write, but as domains become large and grammars approach wide coverage, DCG grammars become less efficient and often harder to maintain than ATNs.

In light of the preceding considerations, THE DIALOG SYSTEM has adopted a Generalized Transition Network parser [Loritz 1993], which is an ATN parser extended by several DCG-like enhancements including shift-reduce mechanisms [Sato 1988], "fitted" bottom-up parsing [Jensen et al., 1993], deterministic "look-ahead" [Marcus 1980], a finite-state morphological transducer [Koskenniemi 1983] which is "cascaded" [Woods, 1980], and a gap-threaded hold list [cf. Alshawi 1992].

The parser currently parses with better than 95% syntactic accuracy on a wide range of constructions including English statements, questions, and imperatives, including passives clauses, inchoative and inceptive clauses, essive clauses, existential clauses, relative clauses, participial clauses and phrases, subordinate clauses, cleft sentences, pseudo-cleft sentences, and various gapped constructions. The grammar therefore meets the definition of a wide-coverage grammar. Parser error, when it does occur, is usually attributable to deficiencies in the lexicon.

The Lexicon

An adequate lexicon is a major problem for all parsing systems. Large, pre-constructed online dictionaries can be adapted for parser use, but in this case there are at least three reasons why bigger, by itself, is not better. First, the lexicon ultimately needs to be coordinated with the ontology, and the larger the lexicon, the more difficult this task becomes. Second, as noted above, homonymy and polysemy seriously degrade a parser's performance. For this reason it is undesirable to have a large lexicon which is deeply coded with many out-of-domain word senses and sub-senses. Third, technical training almost always entails the learning of new technical terms and specialized jargon. As a result, no pre-constructed lexicon, no matter how large, is likely to have the most important words for the (instructional) task at hand.

Instead, our parser operates with a core lexicon of the 4,000 most frequent English words and a main supplementary lexicon of the next-most frequent 16,000 English words [Carroll et al. 1971]. These combine to cover, in appreciable depth, some 30,000 word senses. Additional, domain-specific supplemental lexicons can also be called if an input word is not found in these master lexicons. The lexicon is a neural net in design, but only a semantic net in implementation -- there are no weights yet anywhere on the network nodes.

The upshot of all these design considerations is that lexicon must be authorable. The lexicon was designed for authorability from the ground up. To simplify lexicography, its lexicons were organized as semantic networks: taxonomies into which new terms can be added and from which default lexical features can be automatically inherited [Loritz 1993].

The Representational Hierarchy

The knowledge representation schema is structured as a three-level hierarchy. The base of the representational hierarchy is made up of *alpha-objects*. We have encountered these workhorses of knowledge representation before, under the colloquialism of "factoid". By whatever name, these objects represent simple propositional relationships pertaining among the primitive concepts in the universe of discourse, the equivalent of simple declarative clauses (save that they are composed of language-independent *ideas*, rather than words). Factoids corresponding to "it is raining today" or "Arthur canceled the Sunday-school picnic" typify the KR propositions found at the alpha level. Unlike the building blocks used in some other knowledge representation systems, however, alpha-objects are well suited to modeling propositions about individuals (extensional knowledge) as well as about classes (intentional knowledge), making it possible to override inherited properties and thereby support non-monotonic modes of reasoning. A more extensive discussion of alpha-level factoids may be found in [DeSmedt 1995].

A middle layer of *beta-objects* represents simple, binary relationships between alpha-factoids. One common use for beta-linkage is the "because" conjunction: given the two factoids corresponding to "it is raining today" and "Arthur canceled the Sunday-school picnic," the proposition "Arthur canceled the Sunday-school picnic because it is raining today" is a beta-object.

The apex of the KR pyramid is occupied by *gamma-objects*. They generalize the beta-link structure by accommodating an arbitrary number of alpha-, beta-, or (recursively) gamma-objects in a wide range of relationships -- deterministic and probabilistic logical and relational operators, implication, universal and existential quantification, etc. Since they can quantify over operands which are themselves gamma-relationships as readily as over first-order variables, the expressive power of gamma-objects is equivalent to that of second-order predicate calculus.

It is at the gamma level that the knowledge representation schema attains the ability to capture commonsense propositions like "no one can be in two places at once." More precisely: "for all x, y, z, where (x is a person) and (y is a location) and (z is a location), if (x is at y) and (y is not equal to z) and (y does not contain z) and (z does not contain y), then (x is not at z)" That is, this deceptively simple commonsense rule requires eight atomic propositions, five relational operators (six, counting the "not"), an implication, and three universal quantifications.

Taxonomic Knowledge

Perhaps surprisingly, one aspect of common knowledge that is *not* represented in the CKB — or in any other of the knowledge bases — is taxonomy: the relationship of individuals to classes ("Bob is a golden retriever") and subclasses to their superordinates ("A golden retriever is a dog," "a dog is a mammal"). Such taxonomic relations, dubbed *is-a* links for obvious reasons, are frequently "the most common type of link" in traditional knowledge representation systems [Evett, Hendler, & Spector 1990]. The difficulty is that this architecture compels those KRSs adopting it to commit a goodly proportion of their resources to rediscovering time and again that, e.g., a dog *is-a* mammal.

Our system, on the other hand, employs an explicit *type hierarchy* to represent taxonomic relations. This involves encoding all the information normally contained in *is-a* links into the literal values for the concepts instead. The result is that the class/member relationship between the concepts for "dog" and "mammal" ceases to be an explicit proposition in the knowledge base, and becomes an implicit property of the constant assigned to "dog" in relation to the one denoting "mammal". This results in considerable representational economy, even as it provides universal and existential quantification — along with many of the mechanisms of "inheritance reasoning" — to all intents and purposes *for free*.

Semantic Disambiguation

The parser will disambiguate nouns versus verbs which are the same word – "bat" as a noun, and "bat" as a verb. But it will not [usually] disambiguate "bat" as two different nouns. The parser will not 'know' whether "He saw the bat" is meant as a baseball bat or a winged bat. It leaves this task to be resolved by the present (restrictive) domain.

Using its ontolexicon, MetaLang can filter out parses from EngPars that do not satisfy semantic restrictions on verb cases. Thus while the "bat" is ambiguous in "He saw the bat", the "The bat flapped its wings." is not ambiguous. In such a case, EngPars generates both possible parses (either kind of bat) but MetaLang enforces semantic restrictions as only living creatures typically flap wings so the animal-bat interpretation is the only one that makes sense.

Capabilities and Limitations of the Dialog System

Variability in Understanding

The Dialog system "understands" a range of answers that are semantically equivalent regardless of different wording and syntax, although there are limits to what is handled. It can recognize verbs or nouns as instances of more general categories. Thus it can recognize that a gun is a weapon and answer a question like "Do you have a weapon?" with "Yes, I have a gun." or "Yes, I have a knife."

The Dialog system can resolve the following kinds of variability,

Most **anaphoric** reference including (a) *pronomial reference* ("It beeped") and (b) *sentential reference* ("Yes." "It did."'"It is.") constructions that need to be resolved in terms of antecedent references, principally: personal pronouns (*he, she, it*), propositional anaphora (*I know that*), one-anaphora (*One of our aircraft is missing*), deictics (*this, that, here, there, now, then, you, me*), and definite descriptors (*What did the doctor say then?* referring to a previously mentioned physician).

synonyms (computer=workstation=machine), provided these have been specified in the ontolexicon

hierarchical reference, both for (a) *nouns* (given that "Schmidt has a knife", handles "Does Schmidt have a weapon?") and (b) *verbs* (equating "my machine can talk to other machines" to "my machine can communicate to other machines")

ellipsis, sentence fragments often occurring as requests for an elaboration of the preceding statement (*How?, Why?, With whom?*).

Current Parser and Semantic Limitations

Inevitably there are limitations in what can be handled. Some are due to limitations in the parser (EngPars), some are due to limitations in semantics (MetaLang) and some are due to the interactions between the two.

Concept names must be unique in the ontolexicon for a particular category of speech otherwise confusion between concepts can cause problems when the same word could have multiple word senses. For example, "orange" appears as a noun under "citrus" but does not also appear as a color. It can be added as a noun under color, too. But it is not clear that the parser can distinguish the two senses (e.g., the sentence "The color of your shirt is orange." is proofed to "The shirt's color is an orange."). [note: it does take, "You have an orange shirt"]. Similarly, "tank" appears under a category for armored transportation but not under containers, so the military sense of tank precludes including the water tank sense of tank, at least with the same name and with no possibility of confusion. It could be included in the ontolexicon as the phrase "water tank".

The same word can be used for different concepts if the word is used in different categories of speech. Thus, you can enter *orange* as a noun and as an adjective, or *tear* as a verb (to rip) or as a noun (as from crying).

Prepositional Phrase Attachment. Although the parser can handle individual prepositional phrases, either the parser or the internal handoff to semantics component has problems with multiple prepositional phrases. This is a classic NLP problem. Without considerable context, prepositional phrases are highly ambiguous, as in the sentence *I saw a man on a hill with a telescope.* – do *I, the man,* or *the hill* have *the telescope*?

Examples of sentences not handled:

You are in a tent in the desert in North Africa.
You live in a house by a stream in the forest.

Workarounds:

You are in your tent.
Your tent is in the Sahara desert.
The Sahara desert is in North Africa.

With the second set of 'workaround' factoids, the dialog system should produce correct answers to questions such as "Are you in the Sahara Desert"? "Is your tent in North Africa?"

Cascaded adjectives or adverbs. There are similar problems with multiple adjectives, adverbs, or relational clauses.

Examples of sentences not handled:
You have a heavy steel Japanese sword.
The rat that was on your hat that was on the door ran away.

Workarounds:
You have a heavy sword.
The heavy sword is Japanese.
The rat was on your hat.

Reasoning

Special-purpose reasoning mechanisms are used for efficient reasoning in MetaLang. Inheritance reasoning supports quick determination of hierarchical references by traveling up ontologies from individuals or concepts to more general concepts.

MetaLang uses an underlying clausal form and has reasoning capabilities equivalent to First Order Predicate Calculus. Unfortunately these are not fully available to knowledge base authors due to restrictions on parsing sentences with negation and existentials (see below) and converting the parsed sentences into a logical form for input to MetaLang. Thus, from the viewpoint of a Tutor author, only modus ponens and inheritance is currently available.

Factual Reasoning. The primary kind of built-in reasoning in MetaLang is inheritance. For example, if Fred is a bat, and a bat is a mammal, mammals are warm-blooded, and therefore Fred is warm-blooded. Antecedent reasoning (modus ponens) is also supported. For example, given the factoids below:

Van Gogh is an artist.	artist(van_gogh).
All poor people eat sandwiches.	if poor(x) then eats(x,sandwiches).
All artists are poor.	if artist(x) then poor(x).

MetaLang can deduce poor(van_gogh) and eats(van_gogh,sandwiches) where van_gogh stands for the individual whose name is "Van Gogh".

Current reasoning limitations. Currently, the dialog system does not handle negation and thus cannot handle contrapositive reasoning. For example it does not have the reasoning required to deduce that Bill Gates is not an artist from the following factoids and rules.

Bill Gates does not eat sandwiches.	not(eats(bill_gates,sandwiches)).
All poor people eat sandwiches.	if poor(x) then eats(x,sandwiches).
All artists are poor.	if artist(x) then poor(x).

In an earlier build, Tutor made closed world assumptions. If it did not have a fact in its knowledge base, then it would answer in the negative. For example, if it did *not* have the fact, The Pope is Catholic, then

Tutor would answer the question, Is the Pope Catholic? with "No, the Pope is not Catholic." The handling of such questions has been changed so that Tutor no longer makes a closed world assumption. Now, it would answer, "I do not know if the Pope is Catholic."

Much of what is not handled is due to its restricted set of common-sense rules about word senses and restrictions on inference capabilities. For example, if *x* kills *y* then we (humans) know *y* is dead, but we must add a rule for Tutor to know this, too. In addition to the need for common-sense reasoning rules (a huge project as shown by CycCorp's work on Cyc) there is no inference other than inheritance and modus ponens available to a Tutor author, even though the underlying knowledge representation system MetaLang can handle all the inferences of any first order predicate calculus system.

Examples of Reasoning Not Handled

No reasoning system can handle all kinds of reasoning, including humans. Following is a summary of some of the reasoning that is not currently handled by the Tutor system.

Negation. Currently one cannot author rules or facts that express that something is not the case or never can be. There is some reasoning of this type already encoded as factoids in the Tutor system. For example, there is a built-in semantic transform (procedure) that allows Tutor to deduce that if you hate something you do not like it, and if you are sane you are not crazy. But rules such as, "All poor people are not rich."can not be entered into Common Knowledge.

Existential sentences. Sentences that start with "There is.." are not proofed accurately.

> There is a pink Christmas tree.
> There is a flea on that dog.

Beliefs about propositions. Beliefs about propositions cannot be asserted in sentences. For example, the following cannot be handled:

> This sentence is false.
> John believes everything.
> Mary believes nothing John tells her.

Common-sense Information that Must be Explicitly Entered

The dialog system does not have a large common-sense knowledge base, but it does have an authoring mechanism for this. Until a larger common sense knowledge base is authored, the Dialog system will fail to understand some items that users might expect. Some examples follow: Please note that there is no working system (anywhere) that currently contains all of these types of reasoning and knowledge that human beings use so effortlessly.

Rules about words and what they mean. The meanings of words and their relationships must be explicitly provided. For example, one can author these facts, "You are dead" and "You are alive" and then the character will answer affirmatively to both questions. It has no knowledge that one excludes the other without specific authoring about that fact. There is no information that some states preclude others or that some adjectives imply others

Rules about roles and what they mean. Similarly, the system has no built-in knowledge of social or work roles. For example, it has no knowledge that *x* manages *y* implies that *y* works for *x*, that all wives are women and all husbands are men, that *x* teaches *y* implies *y* learns from *x*, and so on. All such information can be provided with other constraints, but the lack of negation makes it difficult for one term to rule out another.

Temporal reasoning. Although the system has a representation of time, it is fairly primitive. It cannot reason about temporal intervals or whether one interval meets or overlaps another. (You were born in the afternoon. Were you born in the morning?)

Spatial reasoning. The system cannot perform any diagrammatic or map-based reasoning. It needs explicit knowledge that relationships such as "taller" and "larger" are transitive. It does not know how North, South, East, and West are related.

Reasoning about emotions and plans of people. It does not know what motivates people and why they might act the way they do.

Reasoning about natural kinds. The built-in knowledge about natural kinds is limited to what is in Common Knowledge. Particular individuals can be exceptions to the rules (e.g., "Bowser is a dog with three legs.").

Reasoning based on world knowledge. Given "You were born in the afternoon." it cannot answer the question, "Where were you that same morning?" Knowledge about stereotypical situations (eating in a restaurant, childbirth, using an elevator, marriage, death, etc.) must be supplied.

Authoring the Dialog System

In this section we provide an overview of how the Dialog System is authored. Note that authoring and running the Dialog Agents are separate processes. New terms, factoids, rules or changes to a curriculum cannot be provided while in midst of a dialog.

Creating a Dialog Agent

The authoring system allows creation of a new dialog agent from scratch or editing of an existing agent. Currently the system has six character slots available. Any of these slots can be used to create Conversational Agents or Tutor Agents. However, users can only have a dialog with one character at a time.

In the creation of a new character, the author selects the identity elements: title (Dr., Lt.), the first and last name, and the three personality attributes (gender, helpfulness, and manner). The options for the three character attributes are selected via drop-down lists.

Selection of helpfulness and politeness levels affects the dialog characteristics of the agents. Toward the casual-rude end of the spectrum, the agent will assume it is on a first-name basis with its collocutor, or even start insulting them.

Gender —the gender of the agent: male, female, or neuter.
Helpfulness —how cooperative the agent is in answering questions: indifferent, cooperative, uncooperative.

Manner —how polite the agent is to the user in addressing them: obsequious, polite,
normal, casual, impolite, rude.

Adding Knowledge (facts)

The next authoring screen allows facts to be added to the character's own belief set (knowledge base). A fact is entered as a complete sentence. Facts which describe the character itself are given as directions: You believe x, You are y, You like z. You are an artist.

After each sentence is entered as a fact, it is proofed by the dialog system. The proofing process checks to make sure it 'understands' the words and the sentence. It then transforms the sentence to its own format for a factoid. For example, the fact "You are an artist" is transformed to: You are now an artist. The representation that is actually saved 'internally' looks like a case-frame whose slots are populated with numeric concept encodings.

If the dialog system cannot parse the sentences or the questions, or if unknown terms are used, then the dialog system will italicize the unknown text. Sometimes rephrasing the sentence or question will result in acceptance by the dialog system. Individual unknown terms need to be entered into the dialog system (this authoring process is described later).

If we want all Conversational agents and Tutor agents to have access to the same knowledge (facts), then these facts are added to the Common Knowledge base. Providing these facts in the Common Knowledge base does not allow Tutor agents to dynamically incorporate them into new tutorial strategies, but can be drawn upon to formulate responses to trainee's side-bar questions. To employ tutoring strategies, a curriculum needs to be authored for the domain. [This process is described in the section Details of the Intelligent Tutoring System]

Authoring Conversational Agents

To create a Conversational agent, the author creates a new character and assigns it a name and personality attributes. One can then add general or domain specific facts to its knowledge base or to the common sense knowledge base.

Questions that the Conversational agent might pose to the human dialog partner are not authored in question format, as one might suspect, in a form such as "When did you join the Army?" Instead, they are authored as factoids such as "You joined the Army" for the previous example. Any factoid entered into the User KB are flagged with 0 certainty, rendering them as questions for the purposes of language generation.

Extending the Lexicon and Ontology

New topic domains will likely have numerous terms particular to that domain, along with domain-specific facts. For an electronics domain we would need to define semiconductor, resistor, transistor, diode, capacitor, and all the other terms that we would expect to encounter in that domain. Next we

would want to add facts that are the background knowledge for the domain. For example, a transistor always has three leads.

New terms can be entered as individual words first, or the author can choose to author the facts to the knowledge base first. When facts are added, the dialog system performs a check on all terms to see if they are present. If they are not, then it will italicize the unknown term and provide prompts for authoring new terms. If the author suspects that there are many new terms, it will be easier to enter the new terms individually first, and then enter the associated facts.

To add a new term, the author selects its lexical part of speech from a list of alternatives (noun, verb, adjective, adverb). Then the term must be placed under an appropriate category in the ontology. The ontology is a hierarchy of concepts – a large tree-like structure. Each branch and leaf of the tree inherits characteristics of the preceding concept. [see Appendix B: User Manual for examples of authoring screens]. To find the proper category the author searches to see if a relevant concept is already defined. In this case, if a search for "sculptor" shows that this specific term is not in the ontology but the term "artist" is present, then "sculptor" can be placed in the ontology immediately under "artist".

Once a place in the ontology is determined, then the author needs to specify the relationship of the new term to the existing term next higher up in the ontology. Relationship types are: a kind of, a synonym, or a specific individual. For example, a "sculptor" is defined *as a kind* of "artist". Alexander Calder would be *a specific individual* under the class "sculptor" in the ontology.

If there is no logical location in the ontology for a new term, then the author can create a new branch from which to 'hang' the new term.

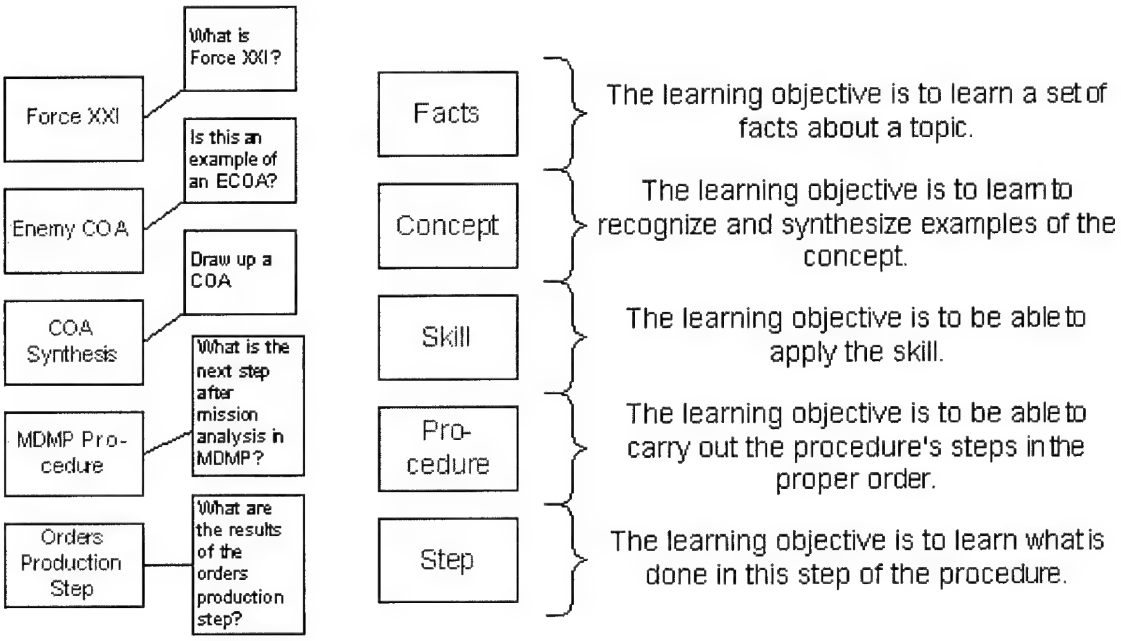
DETAILS OF THE INTELLIGENT TUTORING SYSTEM

This section describes details of the intelligent tutoring system that are normally hidden from casual users. A casual user only needs to author a description of the curriculum to be taught, in a structure called a Curriculum Information Network (CIN), and to select allowable instructional strategies from a pre-stored library. A more sophisticated user can author their own instructional strategies. Each instructional strategy can be applied to many CINs so that different kinds of instructional uses can be provided for the same subject matter.

The CIN describes *what* the material is to teach, but not how to teach it. The instructional strategies decide *how* to teach the material. The instructional plan interpreter interprets an instructional strategy and a CIN to decide what to do next. First we discuss CINs and then how they are interpreted.

Curriculum Information Networks (CIN)

A CIN represents the subject matter to be learned and how the different constituents of the subject matter are related. Each constituent is represented as one of 5 different kinds of nodes: Facts, Concept, Skill, Procedure, or a Step. The figure below provides examples of these different kinds of nodes.



The reason for representing the subject matter in this way is so that different instructional strategies can be applied. Concept may be taught with examples and counterexamples. Procedures may be taught with demonstrations and scaffolded prompting. Facts may be taught with different kinds of Q/A (short answer, T/F, multiple choice, match lists, etc.). Methods for review and query may also differ for different subject matter constituents.

The CIN represents a skill to be learned in a top-down manner augmented with prerequisite links. The most generic kind of node is a Skill node. A skill can be composed of (sub) skills or any other kind of

node. Other nodes are more restrictive. A Facts node can only be further decomposed into other Facts nodes, and a Procedure node can only be decomposed into other procedures and Procedure Step nodes.

The argument for using this kind of representation rests on the Gagne Hypothesis: that we can improve instruction by differentiating different kinds of teaching objectives. The selection of what kind of objectives to represent is based on the work of Merrill's Component Display Theory.

Note that this representation is a compromise. A simpler representation could represent all subparts of the subject matter as Topics, but this would only be appropriate for factual materials. A more complete approach would have representations for motor skills, problem-solving strategies, etc. The compromise selected here is sufficient to handle the target sample domain of Military Decision-Making and can handle many other domains as well.

The key point is that representing the subject matter in a way supports the development of more effective instructional strategies. We discuss instructional strategies next.

Instructional Strategies

An instructional strategy is a particular approach to presenting, reviewing, assessing, or practicing subject matter material. A typical approach for factual material is to first provide an overview of what will be discussed, then to go into detail for each subtopic, and follow up with an assessment.

Many variant strategies can be imagined. A pre-test could always be required. Subject matter material could be motivated by first presenting demonstrations. Review could be top-down, but just covering material that appears not well understood. Both primary instruction and review could require total mastery before moving onto new material. Student requests could be allowed and acted upon with different strategies.

If there were a single strategy that worked for all students all the time for all subject matter then there would be no need to represent multiple strategies. But subject matter differs, students differ, teachers differ in how they would like material presented, and resources differ for presenting materials.

So we would like to present different instructional strategies in a way that makes sense to instructional developers and does not require them to be programmers. The approach used in the Tutor system is to have pre-stored instructional strategies and let the developer choose those that are appropriate.

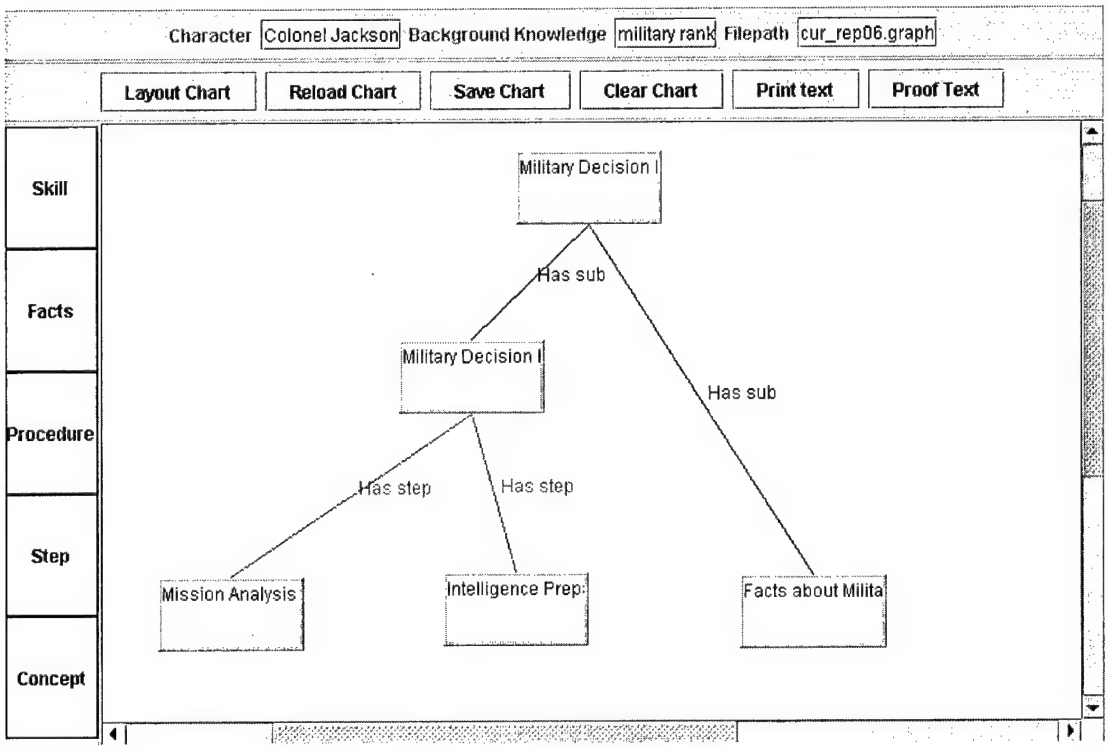
Now we consider the details of how this is implemented. The instructional strategy is represented as a plan composed of steps. The subject matter material is represented as a CIN. The instructional plan interpreter interprets the current step of an instructional plan to decide what CIN topics and what actions to consider. The current step also specifies which actions are preferred and indicates when a step is complete. The plan also has an overall instructional goal (e.g., learn the top-level skill at a better-than-novice skill level with probability greater than 75%).

Currently only two instructional strategies are implemented in the Tutor System. One, for presentation, is called Top-Down Layered Instruction. The other for repair (handling attempts to teach material again when the first attempt failed) is called Top-Down Review.

The idea behind the Top-Down Layered Instruction is that frequently we wish to cover all aspects of a skill in an initial overview before considering details, to better help students see how different parts of a skill relate to each other. If we present too much detail in the beginning the student will get lost.

This instructional strategy is implemented as a plan that traverses through the nodes in a CIN multiple times, each time from top (most generic skill) to bottom (most specific skills or associated prerequisite skills). The first traversal just presents an overview of each topic. The second presents the details.

For example, for the Military Decision Making Process (MDMP) there are 8 steps involved in the full curriculum. The example curriculum stored with Lt. Green is shown below:



The top-level node is Military Decision Making, and is a Skill node. Knowledge about MDMP consists (in this simplified sample curriculum) of procedural knowledge of the MDMP procedure (i.e.., how to do it) along with general factual knowledge about MDMP (e.g., who does it, how to abbreviate it, when to do it). There are actually eight steps in MDMP but we only represent two steps for this sample CIN. The two steps represented are Mission Analysis (MAA) and Intelligence Preparation of the Battlefield (IPB).

Associated with each step in a CIN are teaching actions and learning objects. Teaching actions are actions that can be done for any node, or can occur when a particular node is the activated topic. Narrative teaching actions are the presentation of text for a node. Currently for each node there are three narrative teaching actions: introducing the node, providing a detailed description of the node, and providing a review of the node. Many other kinds of teaching actions could be added in a more complete system, such as simulations, microworlds, and collaborative interactions. Learning objects are specific teaching resources available for particular nodes (skills or topics), such as a graphic or a videoclip. Learning objects are currently implemented as HTML objects, referred to by their URLs.

Unlike traditional computer-based training (CBT), the Tutor system generates its instructional interactions. The interactions are generated by selecting teaching objects and learning objects for

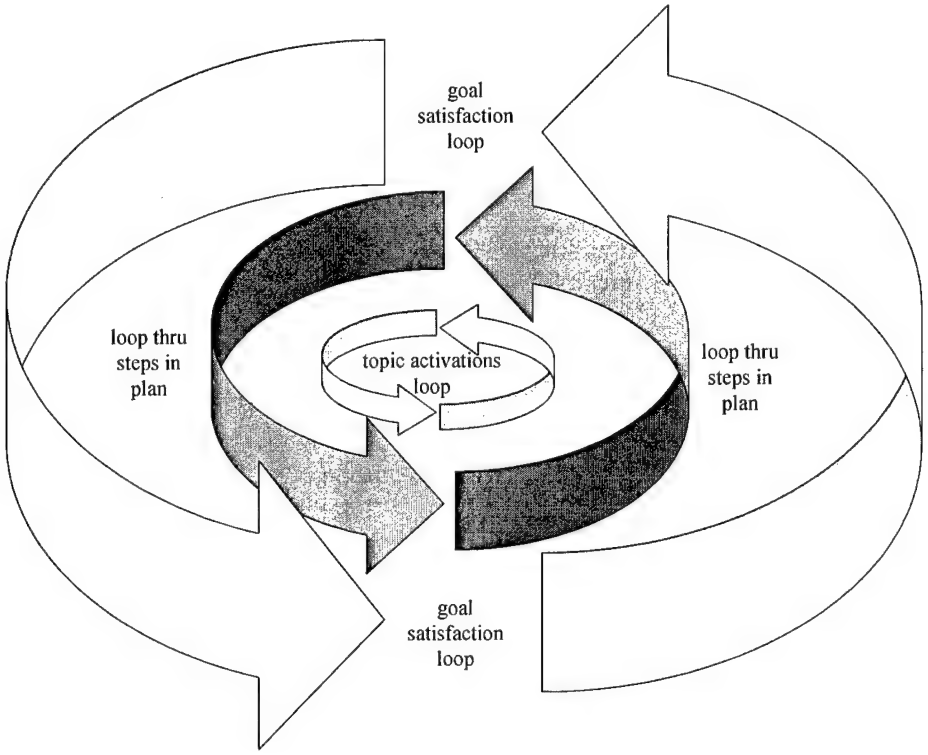
particular topics. The topic selection and choice of actions is handled by the instructional strategy (tutorial plan) and the instructional plan interpreter.

The learning objects introduce a different means of varying instruction that is customized to the student and subject matter. The teaching actions and learning objects together represent what can be done for a particular node (topic or skill). The current instructional step in the instructional strategy represents the actions that are most desired. The instructional plan interpreter chooses the action that best achieves the desired action.

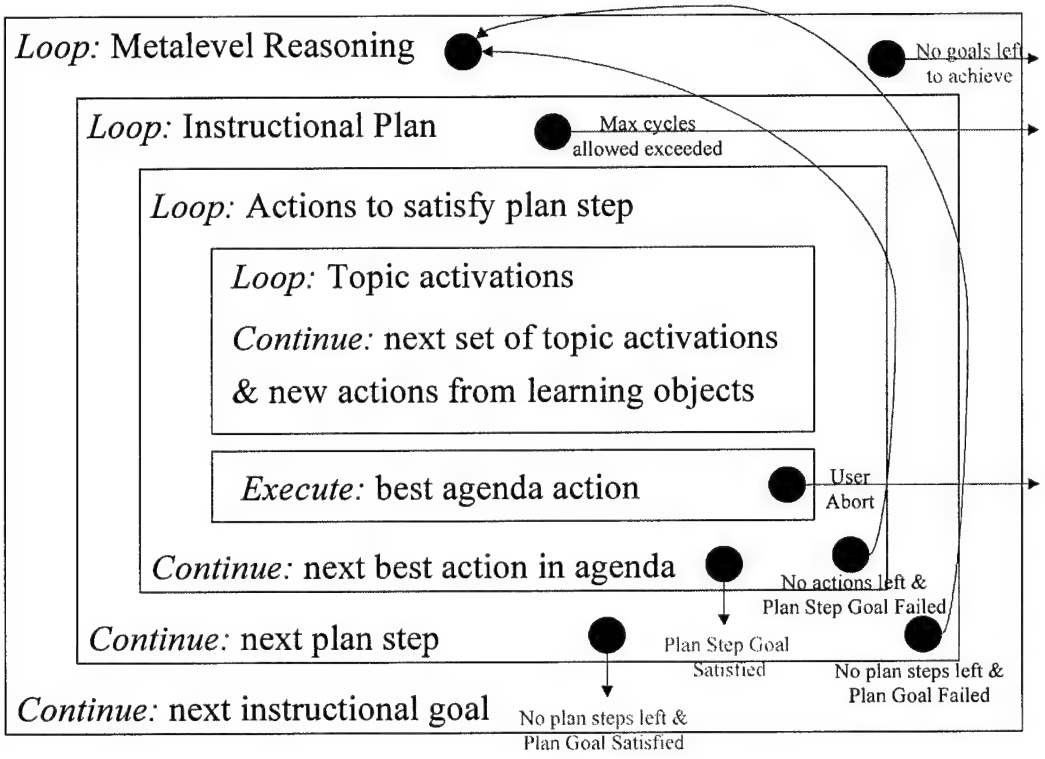
The instructional plan interpreter is a descendant of the BB1 Blackboard Architecture. Is has agendas describing what can be done at each point and control plans that represent what should be done at step in a plan. It has plans that represent preferences and goals that specify when a step and when the overall plan is completed and has a means of handling failed plans.

The next figure, showing embedded loops and arrows, illustrates the overall control of the instructional plan interpreter. At the highest level, the interpreter is attempting to achieve the current tutorial goal, and others may be queued for later achievement or re-achievement. This outermost loop selects one instructional plan for the current objective. The middle loop traverses the steps in that instructional plan. The innermost loop activates topics in the CIN according to a topic-activation policy that is part of the instructional strategy.

A typical topic-activation policy will activate child nodes of a parent topic once the parent topic has been activated. Bottom-up activation policies for review, and random activation policies for testing could also be implemented. Currently the top-down topic activation policy is used in both implemented instructional strategies.



A more typical control-block representation of the instructional plan's actions is shown below. In actual operation exceptions can occur in any loop that may be handled in one of the outer loops. Thus running out of actions in the middle loop may trigger the top loop to choose another instructional plan. Or, if the instructional plan so specified, it could be handled within the instructional plan by choosing proceeding to the next step. But if there is no next step and the plan's goal has not been achieved an exception will be thrown. If one instructional plan fails then another will be tried. An endless cycle is avoided by having the top-level loop limit the number of times an instructional plan can be tried for the same instructional objective.



The instructional plan interpreter is a simplification of the blackboard knowledge bases. Instead of handling any kind of semantic network we use the teaching-specific CIN representation. Teaching actions and learning objects are analogous to domain knowledge sources. The instructional strategies are analogous to control plans but are simpler as separate control knowledge sources are not required to start them, step through them, or to monitor their execution.

Mixed Initiative Dialog

The primary goal of a Tutor agent is to select and interpret instructional plans to teach the top-level objective of the CIN. Each plan step specifies what nodes are active. Teaching actions and learning objects for active nodes are compared to the desired category of action for the current plan step. The best action is chosen, and this process repeats until all topics that can be activated for this step are activated, and either the plan step has been completed or there are no more actions that can be done. In this way a Tutor agent will generate dialog based on the type of currently active node, such as present information, ask questions or evaluate student responses. It will also generate non-dialog actions such as presenting associated learning objects.

Trainees can ask questions of the Tutor agent at any time in a side-bar conversation. The Tutor agent temporarily pauses its tutorial strategy and generates a response to the trainee query. The response is created from relevant facts in its own knowledge base or in the common sense knowledge base. If the trainee stops asking questions, makes a comment, or makes a query to which the Tutor can find no relevant facts, then it resumes the dialog at the last unanswered curriculum node. After completing a curriculum, the user can still carry on a conversation with the Tutor agent about any factoids that reside in its own knowledge base or common sense knowledge base.

Evaluating Student Answers

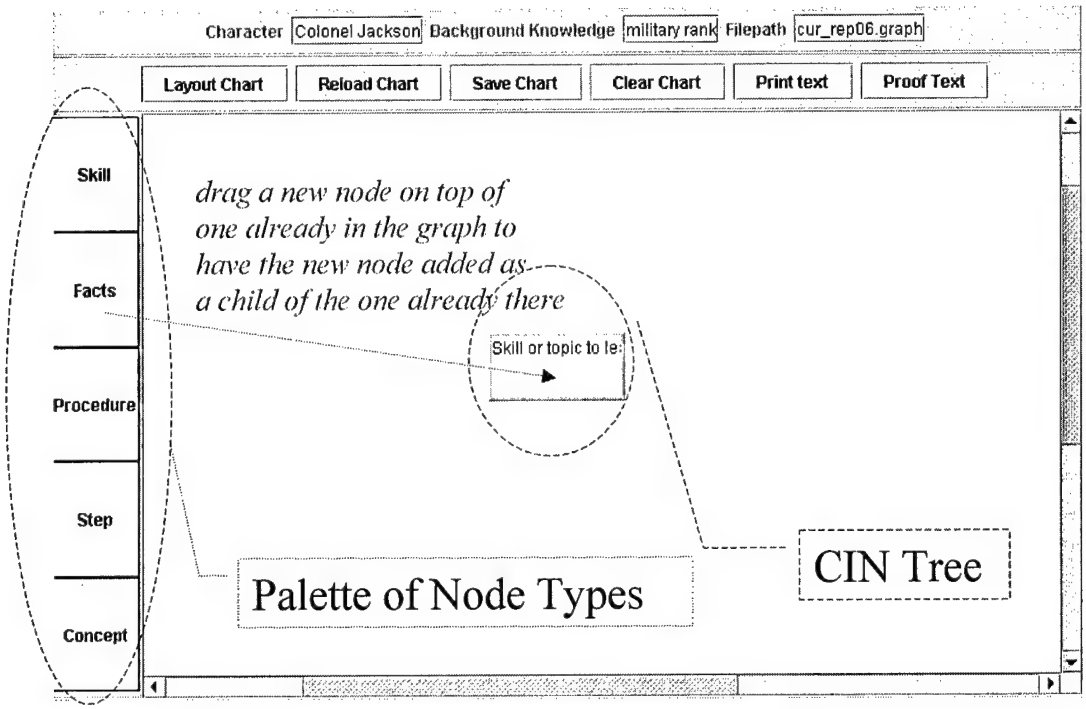
When an assessment question is asked, the Tutor evaluates the trainee's response by comparing it to an exemplar answer that was entered and stored during the authoring process. Answers do not have to match the exemplar word for word, as they are evaluated for semantic equivalence using the dialog system capabilities. They just have to 'mean the same thing' to be evaluated as correct.

The Tutor sends the Dialog system a message containing the query and the user's reply. The message also includes the pre-stored exemplar item answers for the query. The Dialog system compares each exemplar item to the items in the user's reply for semantic equivalence, generality and completeness [this process is described in the section, Dialog System]. The tutor system then receives this analysis and provides feedback on whether the expected items were recognized in the student's answer. It indicates the number of items recognized as correct and the expected items that were not given.

Authoring the Tutoring System

To create a Tutor agent, the author creates a new character and assigns it a name and personality attributes. One can then add domain specific facts to its knowledge base or general facts to the common sense knowledge base at this point, or add these facts after creating the CIN. A CIN is a graphical representation of the target skill to be taught.

The CIN is created in the graphical user interface shown below. On the left is a palette of node types. When a character is first made into a tutorial agent a default graph (the 5-node MDMP) is displayed to show the user what a typical CIN looks like. The user can edit this default graph by right-clicking on nodes to delete them or edit their contents, or by adding new child nodes below existing nodes. The user can clear the graph, starting off with a fresh slate of just one node, as shown in the figure below. New child nodes are made by first placing the mouse over a palette node then clicking on the palette node and dragging it to an existing node, as shown by the dashed blue line in the figure below, used to add a Facts node under the initial Skill node. There are some constraints in constructing the CIN, which are prompted by the system. For example, a Facts node cannot be placed under a Procedure node as only (procedure) Step nodes or Procedure nodes may be placed there.



Each node can have three kinds of narrative properties:

1. An introduction—text to briefly present an introduction to the skill or topic,
2. Details—text to present the skill or topic in detail,
3. A review—text to provide a brief review of the skill or topic.

The authoring system prompts the user for all of the required properties using dialog boxes.

For example:

What name should I give for the topics that these facts address?
D-Day Beaches

An introductory narrative for this topic is _____.
The Allies landed on five beaches in Normandy.

A detailed discussion of this topic is _____.
The five beaches the Allies landed on were code-named Omaha, Utah, Juno, Sword, and Gold.

A short review of this topic is _____.
The Allies picked five beaches for their landing in Normandy.

In addition one or more learning objects may be associated with a node. The user must click on the 'Learning Object' button to pop the next object and look at it in a separate browser window. The GUI could be improved to let the user select from among the queued objects and to indicate the nodes (topics / skills) associated with each learning object.

For example, for the D-Day Beaches node:

Any learning objects?
Yes

OK the URL for this learning object is _____.
http://normandy.eb.com/
[Note: this can be any path designated, does not have to be an http:]

Any more learning objects?
No

Question-answer pairs can be associated with each topic or skill node. There may be any number of these pairs. Dialog boxes are used to prompt the developer for the following parts for each question-answer. While the items are entered one at a time in authoring, the system can handle a user's response with multiple items (compound noun phrase).

Any question answer pairs
Yes

OK the next question to ask is _____.
`What are the code names of the beaches the Allies landed on on D-Day?`

OK the next item should be _____
Omaha

Any more items?
Yes

Note: the items "Utah", "Gold", "Juno", and "Sword" are entered in the same way.

After a CIN is created, the text added for each node is *proofed* (checked to see if the dialog system understands it). The system semi-automates the proofing process by pulling in each node narrative into the dialog authoring system one node at a time. After a curriculum has been successfully proofed then the Tutor agent can provide dialog-based tutoring on the authored domain.

Capabilities and Current Limitations of the Tutoring System

The Tutor system allows a library of instructional strategies to be developed that can be applied to any CIN. The current implementation only has one presentation strategy and one repair strategy. A more complete system would have more instructional strategies in its library. An ideal system would have different kinds and graphical authoring tools for creating new tutorial strategies from scratch.

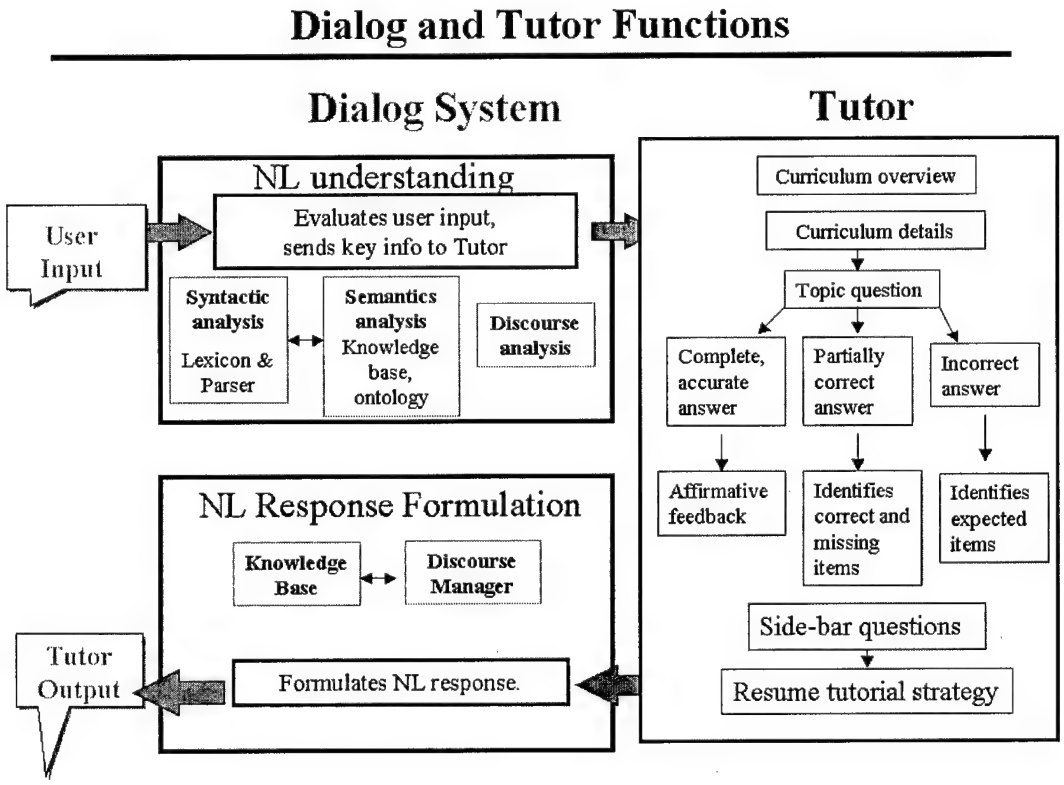
The current system can interpret instructional strategies dynamically for new CINs and for differing combinations of learning objects and teaching actions. Teaching actions in the current system are primarily limited to those available through the Amber NLU system, so these tend to be dialog-specific. But the same controller could operate an intelligent tutoring system without an NLU. For example, it could be used to teach troubleshooting for a complex mechanical device with interactive simulations.

Teaching actions might include highlighting parts, and animating normal and faulted operation. Assessments might include locating parts and predicting fault candidates for symptoms.

Future extensions could be used to specify the instructional plans via a controlled English language interface. For example, a plan might be: Introduce all topics and then present the details of each topic. Query each topic. Practice all exercises for all topics until the intermediate level is reached at 80% probability. Then natural language could be used not only to author domain facts, but also to author teaching strategies.

OVERALL SYSTEM ARCHITECTURE

The Tutor system consists of two main entities. The dialog system and conversational agent facility were created by Amber Consortium. The Tutor agent and overall system control were created by Teknowledge Corporation. The basic concept was to have the dialog system handle the NL interface and the Tutor agent would determine what to do next. Each component communicates with the other.



The Tek Tutor agent controls the tutorial strategy. The Tutor agent is divided into an application server and the curriculum GUI. The latter is only running when a curriculum is being edited. The application server handles the traversal of a curriculum. The Tutor agent also controls all communication between the Tek and Amber agents.

The Dialog system has:

1) The Amber server—an HTTP server that has CGI scripts to intercept messages to the Amber system

2) EngPars—a parser and lexicon for English

3) MetaLang—the ontology and knowledge representation system that handles the ontolexicon and reasoning from rules in Common Knowledge and a character's belief set

4) Dramatist—an authoring system for characters' personalities and beliefs

There is also a launcher to start both the Amber dialog agent and the Tek Tutor agents initially, but this is a simple DOS batch file.

The different parts of the Amber dialog system communicate via shared memory in a DLL (dynamic linked library). The different parts of the Teknowledge Tutor agent communicate via messages sent over sockets. Finally, the Amber agent and the Tek agent themselves communicate with each other via HTML HTTP – messages sent to and received from the Amber server. These messages use a special protocol based on KQML (Knowledge Query Manipulation Language), a commonly used inter-agent communication language.

An HTML front-end is provided for the overall Tutor system GUI. Different paths through that GUI lead to different Tutor capabilities (conversing with a conversation agent, conversing with a Tutor, editing a character's belief set, editing a Tutor agent's curriculum, and extending the ontolexicon). JavaScript is also used in the HTML pages for form-validation, to list options (e.g., different agents that can be talked to), and to pass information (via cookies) to the curriculum authoring tool.

RELATED WORK

One of the key features of the proposed tutor is its support for mixed initiative instruction that mirrors the normal give and take of tutorial dialog. Previous tutors such as the Lower Hoist Tutor (Murray, 1990) or Meno-Tutor (Woolf, 1984) may have simulated such interactions but did not provide natural language capabilities themselves. For example, Meno-Tutor's output could be used as input to the MUMBLE (McDonald, 1983) natural language system but Meno-Tutor itself did not provide natural language output. Our proposed tutor will be directly integrated with the natural language understanding and generation capabilities of an authorable dialog system.

Mixed initiative instruction requires a flexible dynamic planning approach at multiple levels of plan abstraction. Having a plan, or being able to develop a plan once the trainee's needs are known, provides a sense of coherence and global direction. However, rigidly sticking to a plan results in a computer-based training (CBT) system that inflexibly follows a plan to the extent that it cannot accommodate unplanned questions, requests, or changes in goals. Such a system is also harder to adapt to new domains as so many decisions that are made in the plan are specific to the domain, tasks, presentation, curriculum, and expected trainee population.

In general, a balance must be struck between a totally reactive system, such as SOPHIE-I (Brown, Burton, & deKleer, 1982) and CBT systems with immutable plans that can only be followed. Dynamic instructional planning (Murray, 1990) is an approach to controlling intelligent tutoring systems that attempts to strike this balance by providing plans for global coherence, along with additional means for tracking and revising these plans when conditions require changes or allow improvements. The resulting control mechanism results in tutors that are not only more flexible in their interactions with trainees, but also across domains. They are easier to apply to new domains as the pedagogical, assessment, and control knowledge is, by necessity, abstracted into reusable modular units, rather than being procedurally compiled into a human-authored plan, as would be the case of a traditional computer-based tutor.

Dynamic planning allows the tutor to plan incrementally at multiple levels of tutorial abstraction and revise these plans to respond to new opportunities and to the trainee's interactions, including performance, questions, and requests. Dynamic control mechanisms such as MENO-TUTOR have provided similar multi-level plan selection mechanisms, but they lacked the ability to monitor global allocation of resources (e.g., time) and to patch or repair plans on the fly.

Our proposed tutorial system couples, for the first time, dynamic planning, in the form of the instructional plan interpreter, with true natural language processing capabilities. The result should provide a system where lesson planning and discourse planning are truly integrated for the first time. A particular advantage of this approach is that tutorial dialog is well suited to support 'soft skill' domains (e.g., leader training, decision-making). Most intelligent tutoring systems have avoided these areas, as it is easier to build subject matter expert models for precise, formal domains and test trainee knowledge about concrete items such as equipment components and their functions.

REFERENCES

Allen, J. (1995). *Natural Language Understanding* (Sec. ed.). The Benjamin/Cummings Publishing Company, Inc.

Alshawi, H. (Ed.) (1992). *The core language engine*. Cambridge MA: MIT Press.

Bloom, B. S, (1984). The 2-sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher, 13(6),* 4-16.

Brown, J.S., Burton, R.R., and de Kleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II, and III. In Sleeman, D.; and Brown, J.S. (Eds.), *Intelligent Tutoring Systems*. New York: Academic Press.

Carbonell, J.R. (1970). *Mixed-Initiative Man-Computer Instructional Dialogues*. Unpublished doctoral dissertation, MIT, Cambridge.

Carroll, J.B., Davies, P. & Richman, B. (1971). *The American heritage word frequency book.* Boston: Houghton Mifflin.

Collins, A. (1975). Analysis and synthesis of tutorial dialogs. In Bower, G. (Ed.) *The Psychology of Learning and Motivation (Vol. IX).* New York: Academic Press.

DeSchmedt, W. (1995). H. Herr Kommisssar: An ICALL Conversation Simulator for Intermediate German. In Holland, M., Kaplan, J. and Sams, M. (Eds.) *Intelligent Language Tutors: Theory Shaping Technology,* 153-174. Lawrence Erlbaum Associates, Publishers.

De Kleer and B. C. Williams (1987). Diagnosing Multiple Faults. *Artificial Intelligence*, 32, (1), 97-130.

Extempo. www.extempo.com

Hendrix, G., Sacerdoti, E., Sagalowicz, D., and Slocum, J. (1978). Developing a natural language interface to complex data. *ACM Transactions on Database Systems* 3, 2:105-147.

Jensen, K., Heidorn, G, & Richardson, S. (1993). *Natural language processing: the PLNLP approach.* New York: Kluwer

Koskenniemi, K. (1983). Two-level model for morphological analysis. *Proc. IJCAI*, 683-5.

Lenat, D. B. (Sep 1995). Artificial Intelligence. *Scientific American.*

Lesgold, A., Lajoie, S. P., Bunzo, M., & Eggan, G. (1992). A coached practice environment for an electronics troubleshooting job. In J. Larkin, R. Chabey, & C. Cheftic (Eds.), *Computer assisted instruction and intelligent tutoring systems: Establishing communication and collaboration*, 201-238. Hillsdale, NJ: Lawrence Earlbaum Associates.

Loritz, D. (1993). Generalized transition network parsing for language study: The GPARS system for English, Russian, Japanese and Chinese. *Calico Journal 10*, 1: 5-22.

Loritz, D. (1995). GPARS: A Suite of Grammar Assessment Systems. In Holland, M., Kaplan, J., and

Sams, M. (Eds.), *Intelligent Language Tutors: Theory Shaping Technology*, 121-133. Lawrence Erlbaum Associates, Publishers.

Marcus, M. (1980). *A theory of syntactic recognition for natural language*. Cambridge, MA: MIT Press.

McDonald, D. (1983). Natural language as a computational problem: an introduction. In Brady, M. & Berwick, R. (Eds.). *Computational models of discourse*. MIT Press. Cambridge, MA.

Murray, W.R. (1990) A Blackboard-based Dynamic Instructional Planner. In *Proceedings Eighth National Conference on Artificial Intelligence*, 434 - 441. AAAI Press / MIT Press.

Pearl, J. (1997) Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference. Morgan Kaufmann Publishers. Paperback edition of original 1986 hardback.

Platt, C. (April 1995). What's it mean to be human anyway? *Wired.com* http://www.wired.com/wired/archive//3.04/turing.html?person=alan_turing&topic_set=wiredpeople

Sato, P.T. (1988). A common parsing scheme for left- and right-branching languages. *Computational Linguistics 14*:1, 20-30.

Towne, D.M., Munro, A., Pizzini, Q.A., and Surmon, D.S. (1987). Simulation composition tools with integrated semantics. *Abstracts of the Third International Conference on Artificial Intelligence and Education*, 54. Learning Research and Development Center, University of Pittsburgh.

Wenger, E. (1987). *Artificial Intelligence and Tutoring System: Computational and Cognitive Approaches to the Communication of Knowledge*. Morgan Kaufmann.

Woolf, B.P. (1984). *Context-dependent planning in a machine tutor*. Doctoral dissertation, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts.

Woods, W.A. (1980). Cascaded ATN grammars, *AJCL 6*: 1, 1-12.

Appendix A: Sample Tutorial Dialog

Lt Green starts with an overview of MDMP:

MDMP consists of two steps: mission analysis and intelligence preparation of the battlefield.
MDMP is the army's step-by-step approach to making plans to achieve missions.
Intelligence preparation of the battlefield, or IPT, is an analysis of …

Next, Lt Green presents the details:

MDMP is used to choose the best COA considering all METT-TC factors, including possible enemy COAs.

IPB determines the commander's critical intelligence requirements (CCIR). One output is a modified combined obstacle overlay (MCOO), which can be used to determine friendly and enemy information requirements, terrain, high value targets, expected avenues of approach, etc.

Mission analysis determines what the requirements are for the mission.

A complete MDMP has eight steps. The steps can be abbreviated,….

after presenting all material then Lt Green evaluates the trainee's understanding…

Green: Which steps in the MDMP process result in warning orders?
Trainee: MAA and IPB.
Green: 1 correct, 2 missed. Mission analysis is correct, but I also expected: receipt of mission, and COA approval.

Note that MAA was accepted as a synonym for Mission Analysis.

Continuing, here is an example of a sidebar conversation…

Trainee: What is a COA?
Green: Course of analysis.
Trainee: How long have you been in the Army?
Green: I have been in the Army four years.

When the trainee no longer asks questions, then Green resumes the tutorial.

Green: What are the steps to IPB input?
Trainee: landscape map, target analysis

Note that landscape map was accepted as semantically equivalent to terrain map [the stored exemplar].

Appendix B: User Manual

This user manual describes how to have conversations with tutorial and conversational characters in the Tutor Dialog System, and how to author new characters and new instruction for additional subject matter domains.

Overview

The Tutor Dialog System (hereafter just "Tutor") is a prototype natural language tutor that provides simulated characters that can talk with the user. These conversations can be tutorial dialogs, where the character plays the role of a tutor, introducing new material and asking questions, or the conversations can be more interactive, where the student asks the tutor questions.

When a character acts initially as tutor it is called a *tutorial agent*. When it acts primarily to answer user questions it is called a *conversational agent*. The tutorial agent can answer student-initiated questions and the conversational agent can initiate questions, too, but the key difference is that the tutorial agent initiates most of the questions whereas the conversational agent does not.

The limitations of grammar and understanding for both kinds of agents are presented in detail in the final report. Here we concentrate on using the system. We also assume the system has been properly installed as per the Installation Instructions.

Launching and Quitting

To launch Tutor double-click the Tutor shortcut on your desktop. The calls the batch file c:\tutor\tutor.bat, which launches the various programs that are part of the Tutor Dialog System (parser, authoring or conversation support, dialog and top-level GUIs, etc.). Wait for the top-level screen to be presented for Tutor. This should take about a minute on or less on a fast machine (500 Mhz or better, 256 MB RAM).

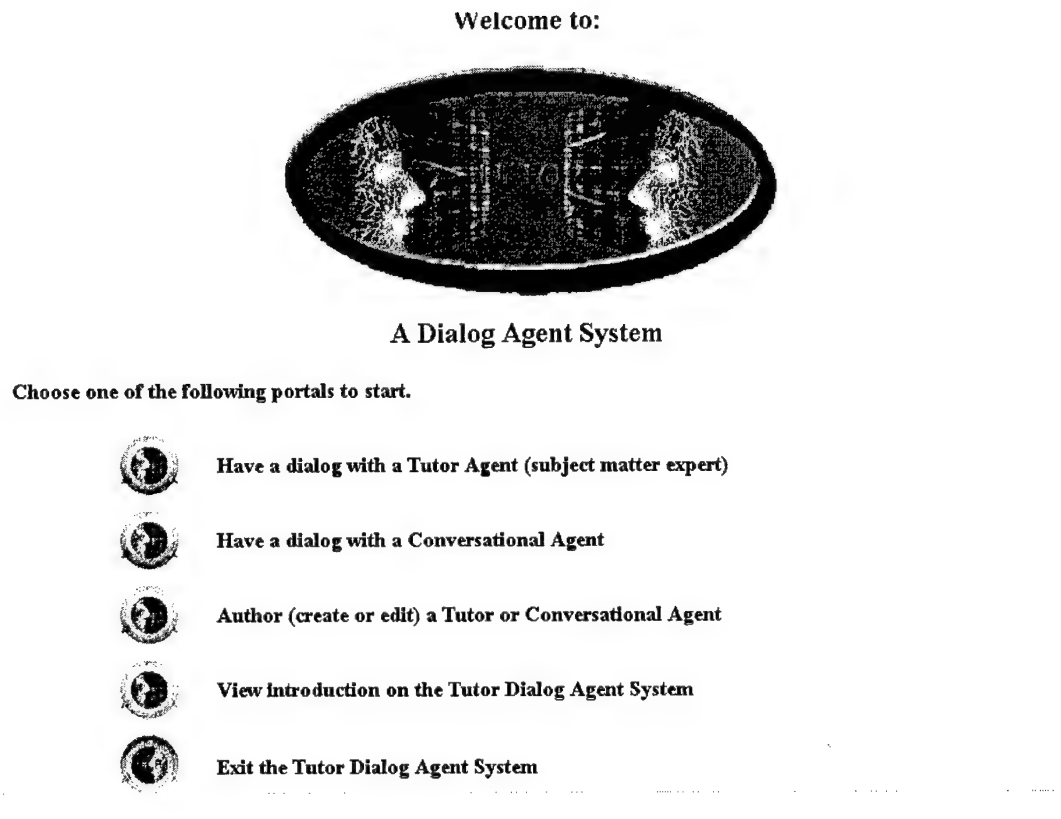
To quit anytime from the top-level screen select the last option "Exit the Tutor Dialog System". That provides a clean shutdown of all the Tutor processes.

Choices from the top-level screen

The top-level screen provides 5 options:

1. Have a dialog with a Tutor Agent (subject matter expert)—allows you to choose from any of the characters that are tutorial agents for a short tutoring session on their area of knowledge.
2. Have a dialog with a Conversational Agent—allows you to ask questions of a conversational agent.
3. Author (create or edit) a Tutor or Conversational Agent—leads to any of the three types of authoring: OLA authoring (extending the lexicon), CIN authoring (defining new topics and skills to teach), or character authoring (defining new characters and what they believe).
4. View introduction on the Tutor Dialog Agent System—shows a short introduction of the Tutor Dialog Agent System and a sample dialog.
5. Exit the Tutor Dialog Agent System—exit the tutor dialog agent system.

Here is a snapshot of the top level screen:

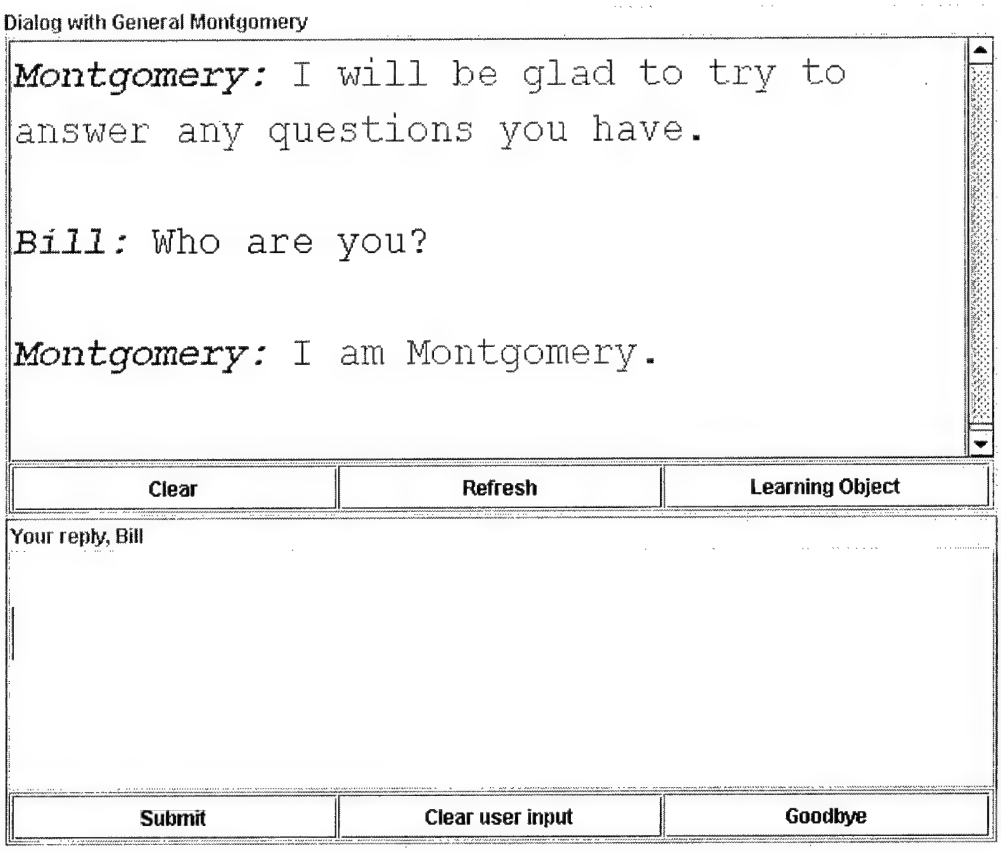


Dialog with a conversational character.

Select the character you wish to talk to from the list of characters on the next screen. Note that all characters that have been authored will appear since all characters are conversational characters although not all are tutorial agents.

After pressing the Next button the character will be launched and you can ask questions. Here is a sample screenshot talking to the character Montgomery:

Conversational Agent Dialog Window



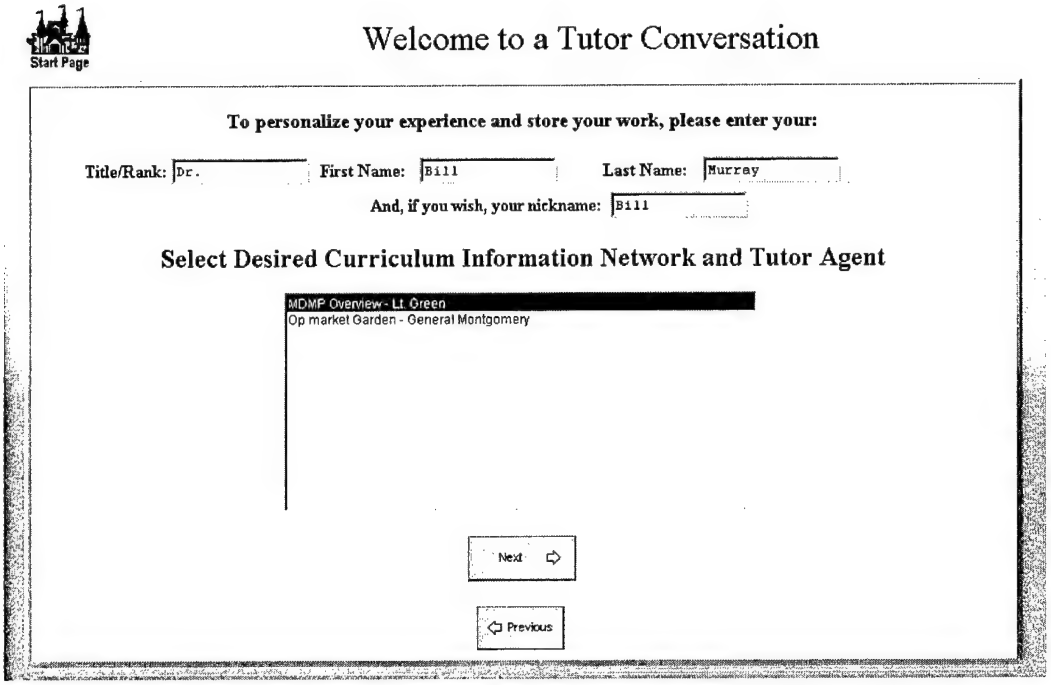
Type questions, ending with a question mark, in the window that is labeled 'Your Reply'. Hitting the Enter key is the same as pushing the 'Submit' button. The button 'Clear User Input' clears the dialog in the window above. When you are through talking to the character select the 'Goodbye' button.

Even if no facts have been entered into a character's set of beliefs ('factoids') you can always get an answer to certain questions such as 'Who are you?' and 'Where are you?' When specific facts or background knowledge is present these, too, can be queried.

Bill: What do you have on your uniform? Montgomery: I have stars on uniforms of a general.

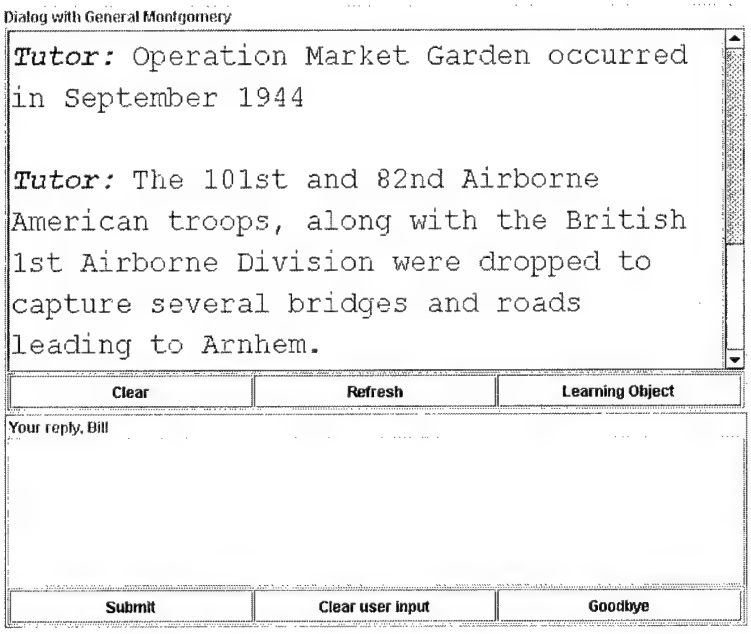
Dialog with a Tutor

A tutorial agent is engaged similarly. The list of tutorial agents appears, as in the screenshot below:



The dialog window is similar to the one shown before, but now the tutor agent will present new material to you, as shown below:

Tutor Dialog Window



This kind of agent is also more likely to ask *you* questions. The tutor may ask a question to which one or more items will be the correct answer. Just list the items as shown below for the sample Q-A interchange:

Tutor: Which armies landed in Normandy?

User: The American Army, the British, and the Canadians.

Note do *not* answer in any other format, such as these formats:

Complete sentences: The American Army, the British, and the Canadians landed at Normandy.

Multiple sentences: The Americans. The British. The Canadians.

Itemized lists: (1) Americans (2) British (3) Canadians.

Tutor does evaluate your answers for semantic equivalence to stored answers, so you can use acceptable abbreviations (defined in OLA) and variant syntax within the noun phrases, one for each item.

```
The Army of America, UK Army, and the Canadians.
```

Do not use additional punctuation other than comma (,) period (.) and question mark (?). Specifically do not use possessives and do not use periods in acronyms (use USA but not U.S.A.). Question marks can be used in answering a question with a sidebar question (one that the tutorial agent should be able to answer), such as, What is D-Day? but not questions such as 'Why do you ask?' which it will not be able to answer.

Asking questions (sidebar conversations)

After answering a sidebar (digressionary, user-initiated) question the tutor will return to its previous question:

Tutor: Which armies landed on D-Day?
User: When was D-Day?
Tutor: D-Day was June 6th, 1944.
Tutor: Which armies landed on D-Day?

Displaying Learning Objects

To activate a learning object, click on the 'Learning Object' button. The number of queued learning objects will be displayed. Select OK to the dialog box and to the next (the object's URL) then the object (e.g., a web site, image, or any HTML object) will be displayed in a new window that appears in the browser. For example, there is one learning object associated with Montgomery's instruction on Operation Market Garden. It is a web site. However, objects can be located in any path, does not need to be a web site.

Authoring

There are three different kinds of authoring in Tutor:

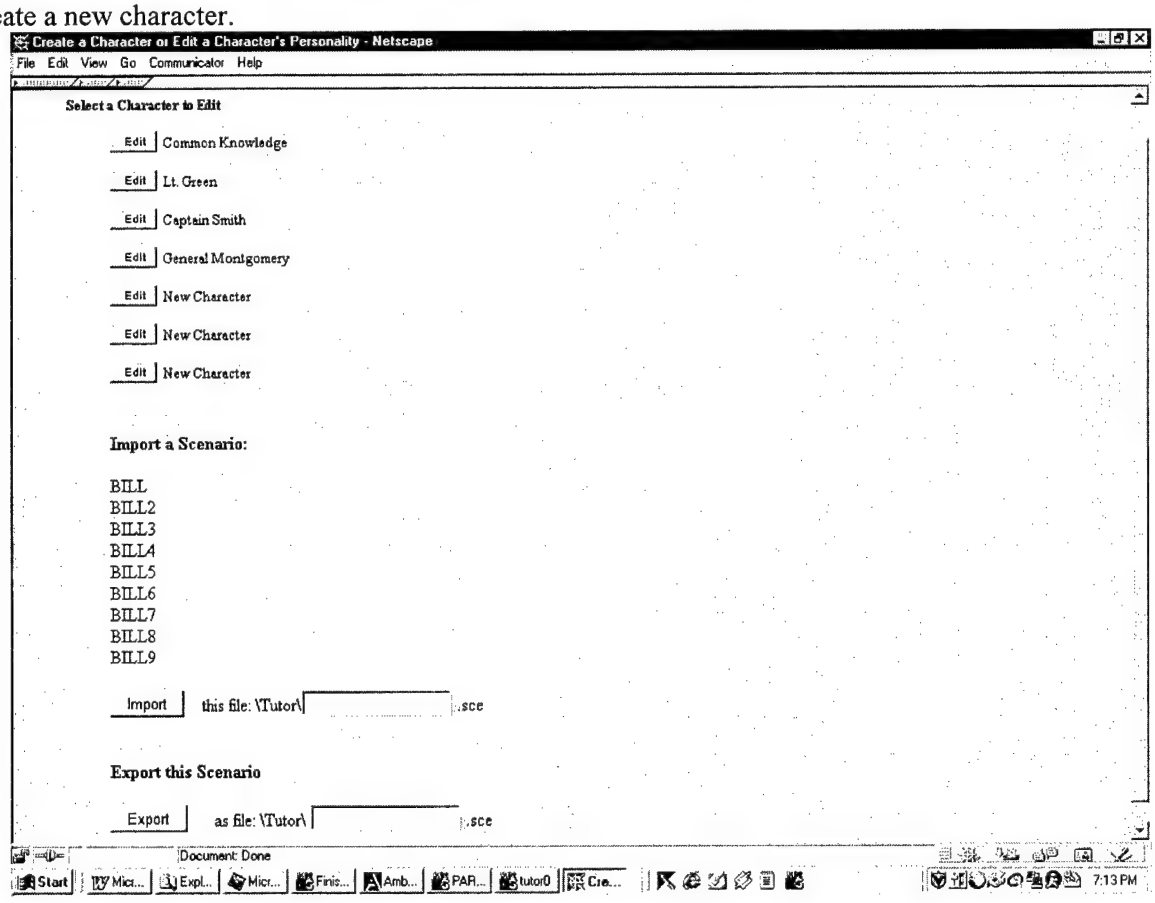
1. Character Authoring—allows new characters to be defined with different parameters for their personalities. Each character can have a different set of beliefs. This set of beliefs, the 'factoids' that the character believes, are the knowledge that is accessed in sidebar conversations with tutorial agents, or in dialogs with conversational characters. Also, there is a special set of beliefs called 'Common Knowledge' that all characters access in addition to those specifically defined for them.

2. OLA Authoring—OLA stands for 'ontology and lexicon authoring'. This kind of authoring adds new words for new subject matter. The lexical aspects of the words (e.g., 'noun') and the ontological aspects of the words (e.g., what category of object a new noun belongs to) must both be specified in terms that are predefined.

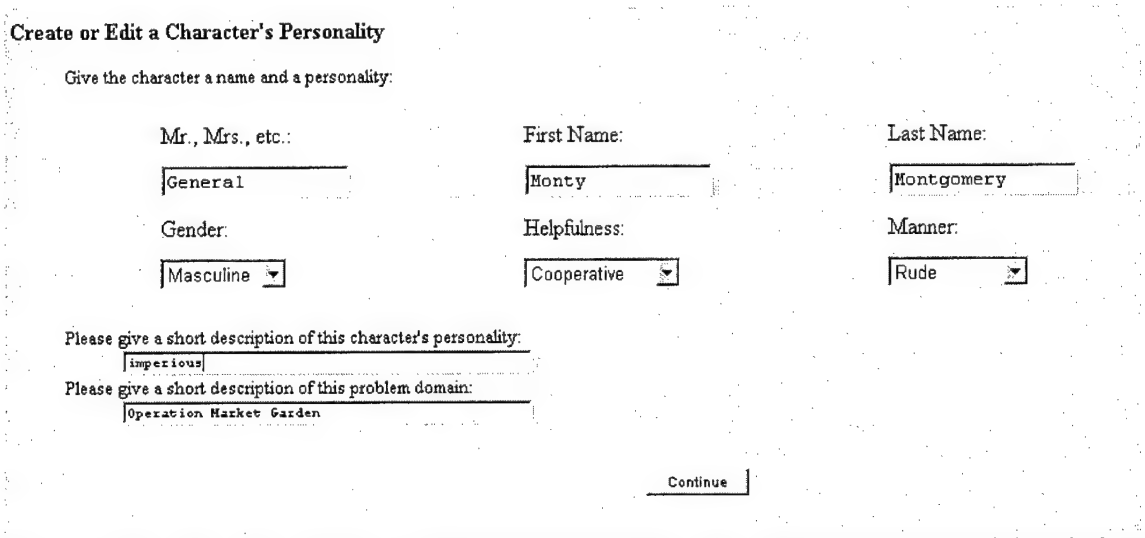
3. CIN Authoring—CIN stands for 'curriculum information network'. This kind of authoring defines new topics and skills and their pedagogical relationships (e.g., one topic is a prerequisite of another). The knowledge authored is pedagogical knowledge since Tutor accesses it in deciding what topics to present next, how to present them, and how to assess what the student has learnt about them. Only tutorial agents have CINs.

How to add a new character

To define a new character go to the top-level screen and select the 3rd (authoring) option. First you will be taken to a screen where you can select the character to edit or one of the 'New Character' slots to create a new character.

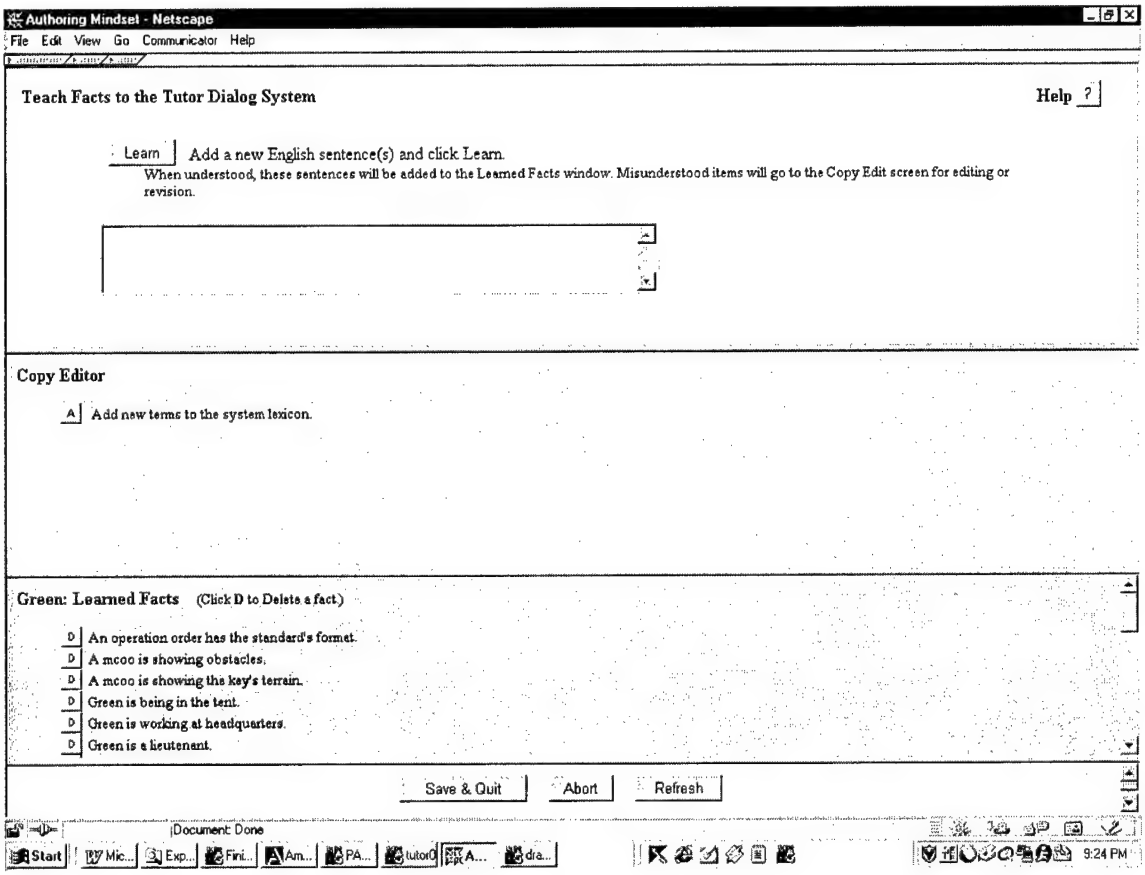


Once you have selected a character you will be presented with a character editing screen, such as the one shown below,



where you can set the parameters for a character's personality and can provide a general description of their particular area of knowledge.

Once you have selected parameters for the character's name, personality, and area of knowledge, then select 'Continue'. Within about a minute you should be presented with the OLA Authoring Screen. This screen lets you enter facts the character knows. It also allows you to add new words to the OLA system.



you would type new facts, such as 'You are a general.', into the first text box, then push the 'Learn' button, to add new facts. We'll talk more about OLA authoring—adding new words to the lexicon and ontology—a bit later on.

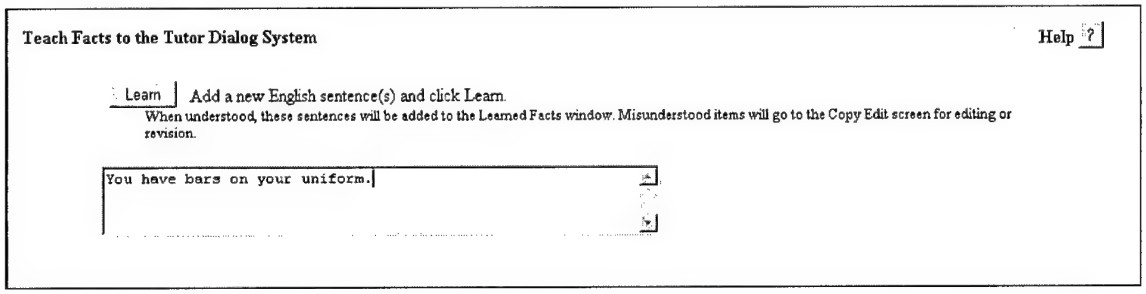
Adding new beliefs for characters or background knowledge

Enter each fact that the character should know. Use simple sentences and phrase your sentences in this manner 'You…' as 'You are a general.' or 'You command 120 tanks.' Note that sentences beginning with 'I' are not accepted (e.g., *not* 'I am a general.').

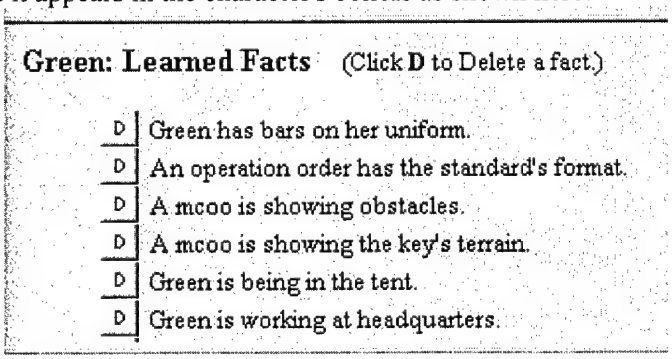
After each fact is typed in the box under the 'Teach Facts to the Tutor Dialog System' label, push the 'Learn' button. You will have to wait about a minute or two then the authoring system will either accept the sentence as is, or will request rephrasing.

If accepted a new fact will appear in the character's beliefs. The new fact may be reworded as a paraphrase showing how Tutor interpreted your statement. If the meaning is different from what you intended then push the D button next to the fact to delete it, and try reentering it another (usually simpler) way.

For example, if we tell a character Lt. Smith, 'You have bars on your uniform', the entry of the fact is shown below:



and after being accepted it appears in the character's beliefs as shown here:



If a sentence is not accepted one or more unrecognized words may be italicized. In that case either define the words as described below under OLA authoring or rephrase the fact to avoid the unknown word(s).

How to save a scenario

Scenarios can only be saved from the same screen as that used to select characters for editing. At the bottom of that screen enter a new scenario name (e.g., 'DEMO') in the box to save a scenario as a file DEMO.SCE.

A scenario saves all characters, the lexicon, the ontology, and all CINs. Essentially everything is saved except you can now save state in the middle of a conversation.

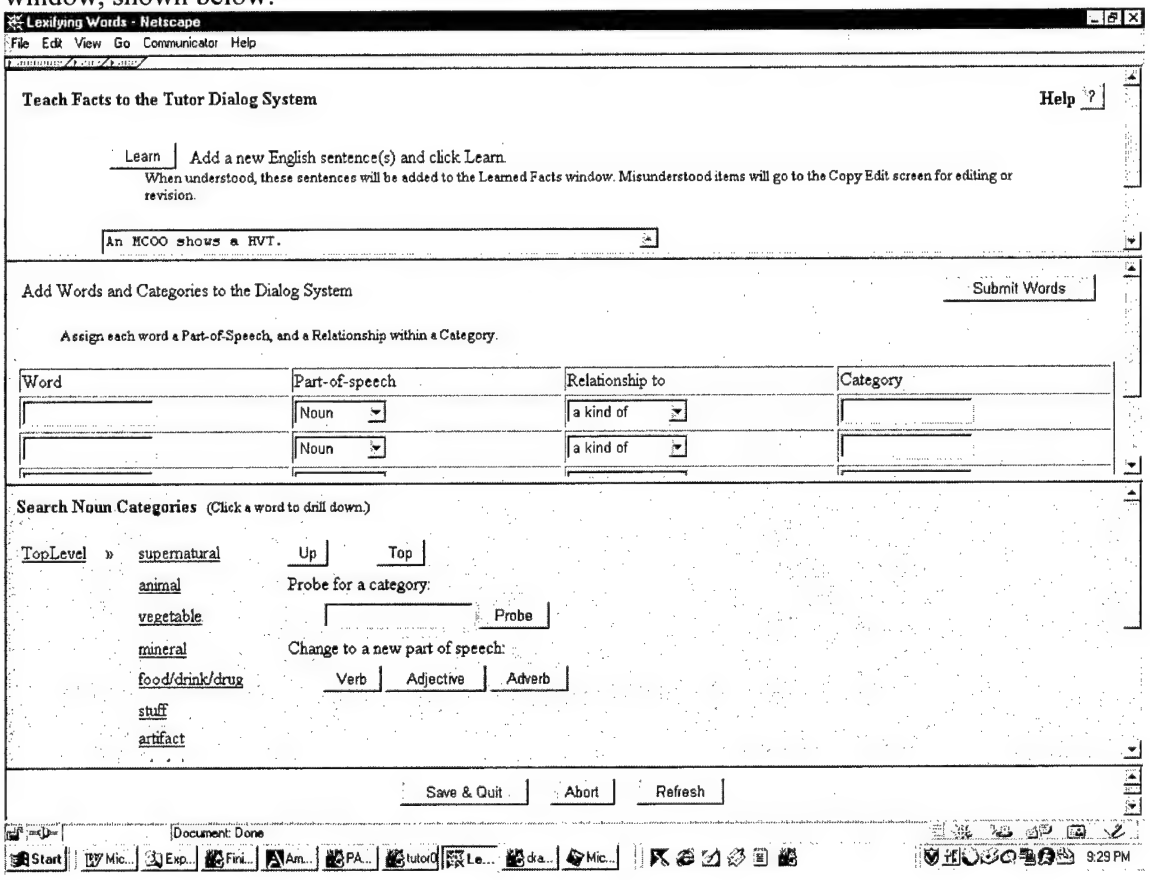
How to load a scenario

The same screen is also used to load a scenario. Either type in the name of a scenario (e.g., 'DEMO' to load 'DEMO.SCE') or use cut and paste from the list of previously stored scenarios above.

Tutor will uncompress information from the scenario file and save it in the appropriate files in Tutor's file system. Be careful that you first save any information you want to preserve before loading a scenario as all previous information is overwritten.

Defining new words and concepts (OLA authoring)

Now we discuss OLA authoring. All OLA authoring is accessed from the OLA Authoring Screen, which appears when entering character beliefs. Thus you must be editing a character to define new lexical and ontology terms. Select the button 'Add new terms to the system lexicon' to bring up the OLA authoring window, shown below:



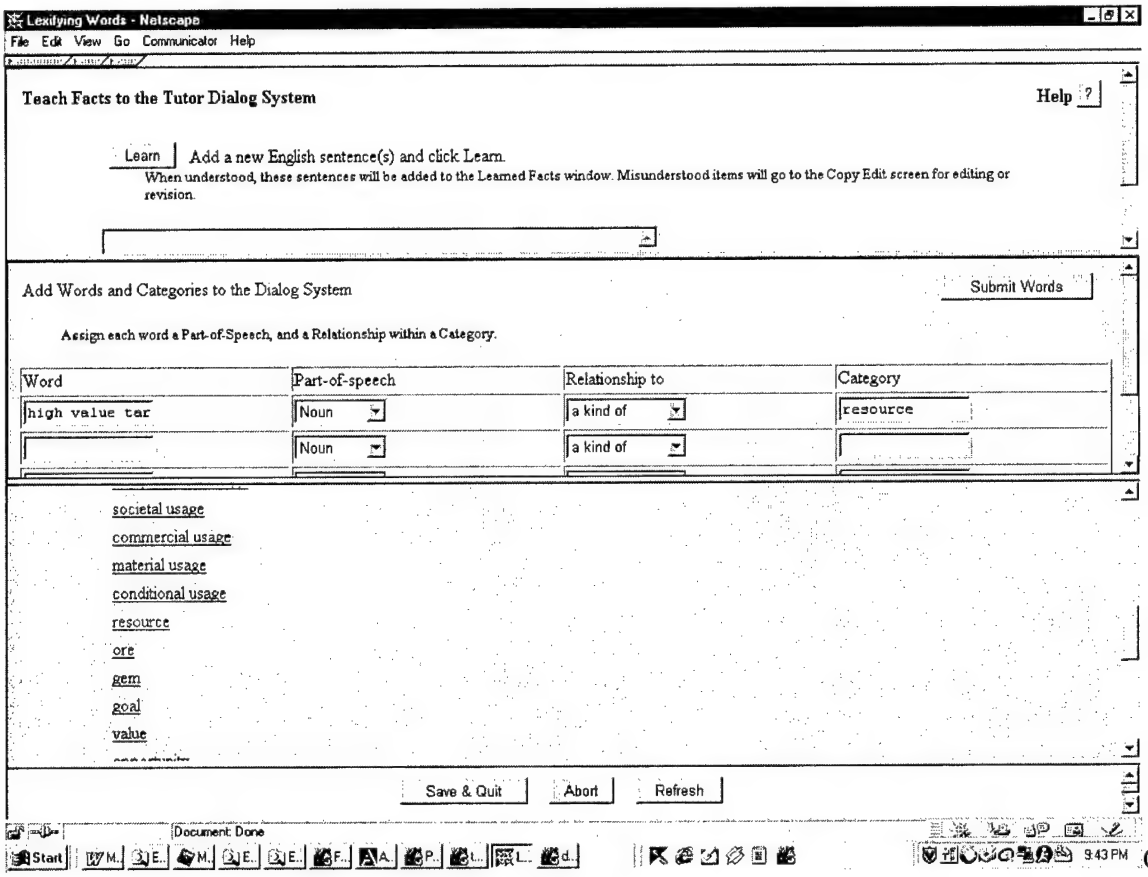
Adding new categories

To add a new category it must be placed below an existing category. So how do you know what category to place it under? You have to search the existing ontology. To do this you probe for the category.

For example, suppose we want to enter the fact 'An MCOO shows all HVTs.' meaning that an MCOO (modified combined obstacles overlay) shows all HVTs (high-value targets). Assume an MCOO is already defined as a map overlay but HVT is not defined.

We will define HVT as a synonym for 'high-value target' but first we have to define a high value target. We probe for 'target' and find that it is not part of the ontology. We try 'goal' and see 'resource' listed nearby, and then decide 'resource' may be better than 'goal' (e.g., an enemy headquarters can be a HVT, or any other enemy resource). But the key point is that first you must search for a category to extend and decide on one.

Next, we define 'high value target' as a noun that is *a kind of* 'resource', as shown here:



OLA accepts this addition and acknowledges the change…

Copy Editor

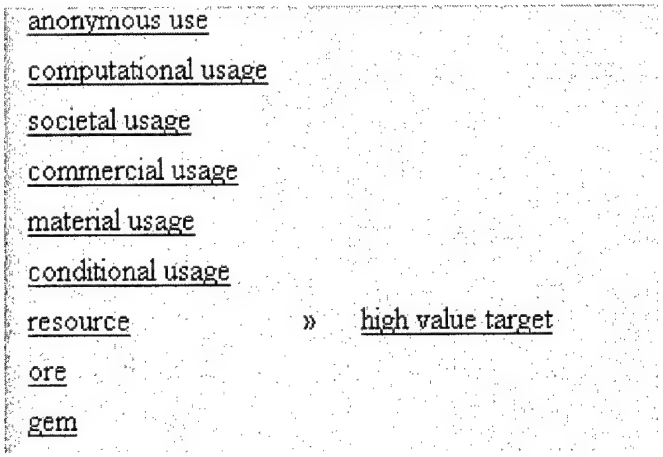
The term

high value target

was successfully learned.

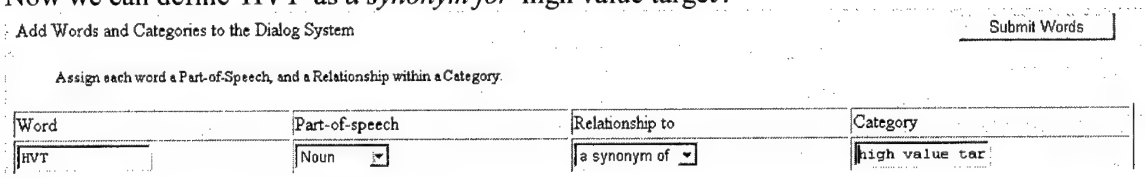
A Add new terms to the system lexicon.

We can also probe for 'resource' now and see that 'high value target' is listed below it in the ontology, as shown below:

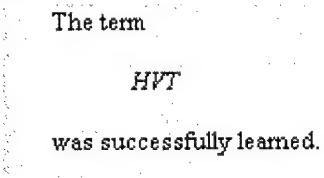


Adding new synonyms

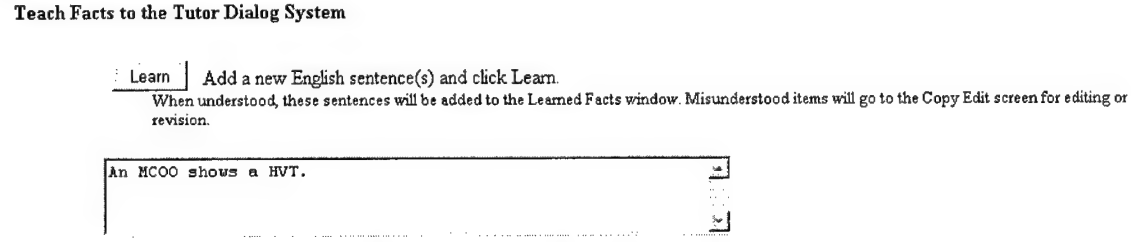
Now we can define 'HVT' as *a synonym for* 'high value target':



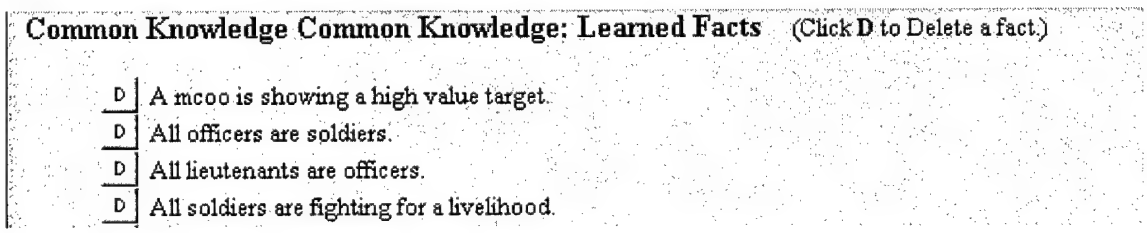
OLA acknowledges this addition, too:



Finally, we can enter the fact 'An MCOO shows a HVT.' as shown below:



OLA shows that this sentence is interpreted as 'An MCOO is showing a high value target'.

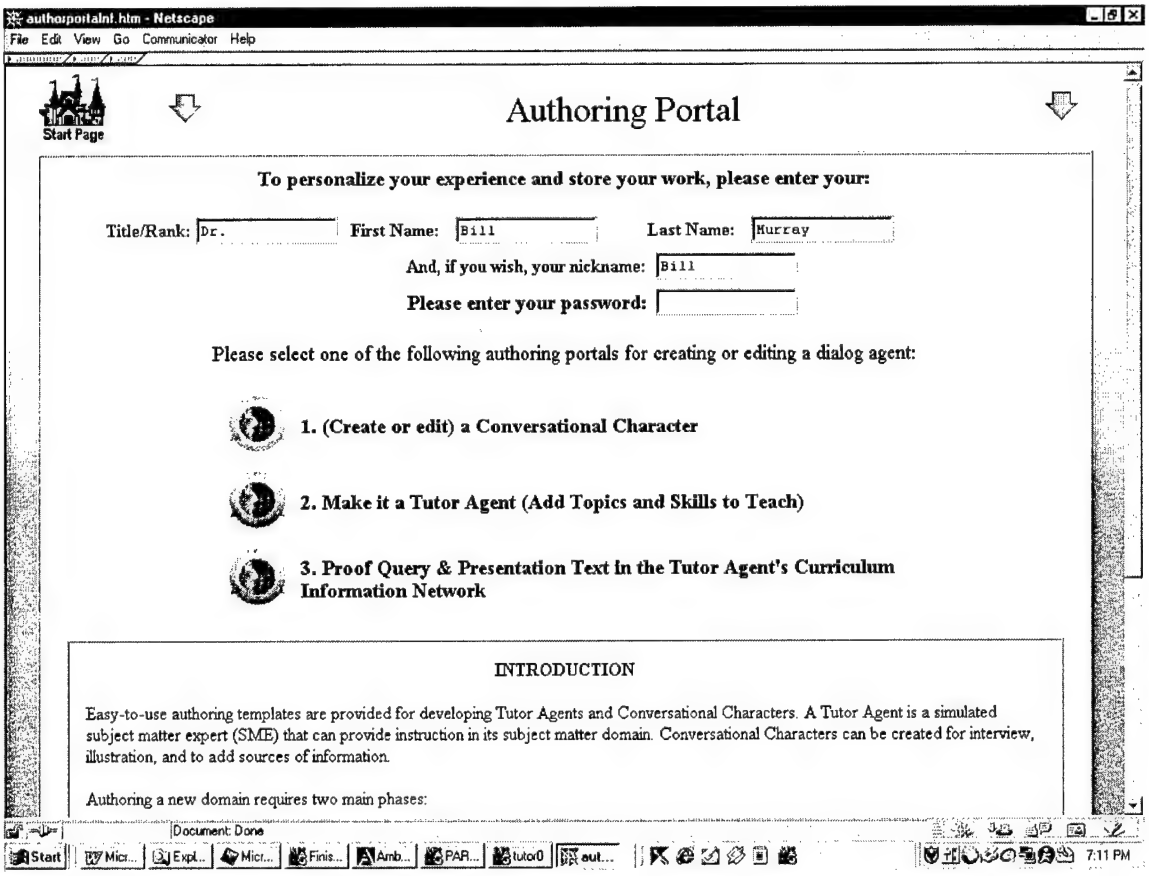


How to change a character to a tutorial agent

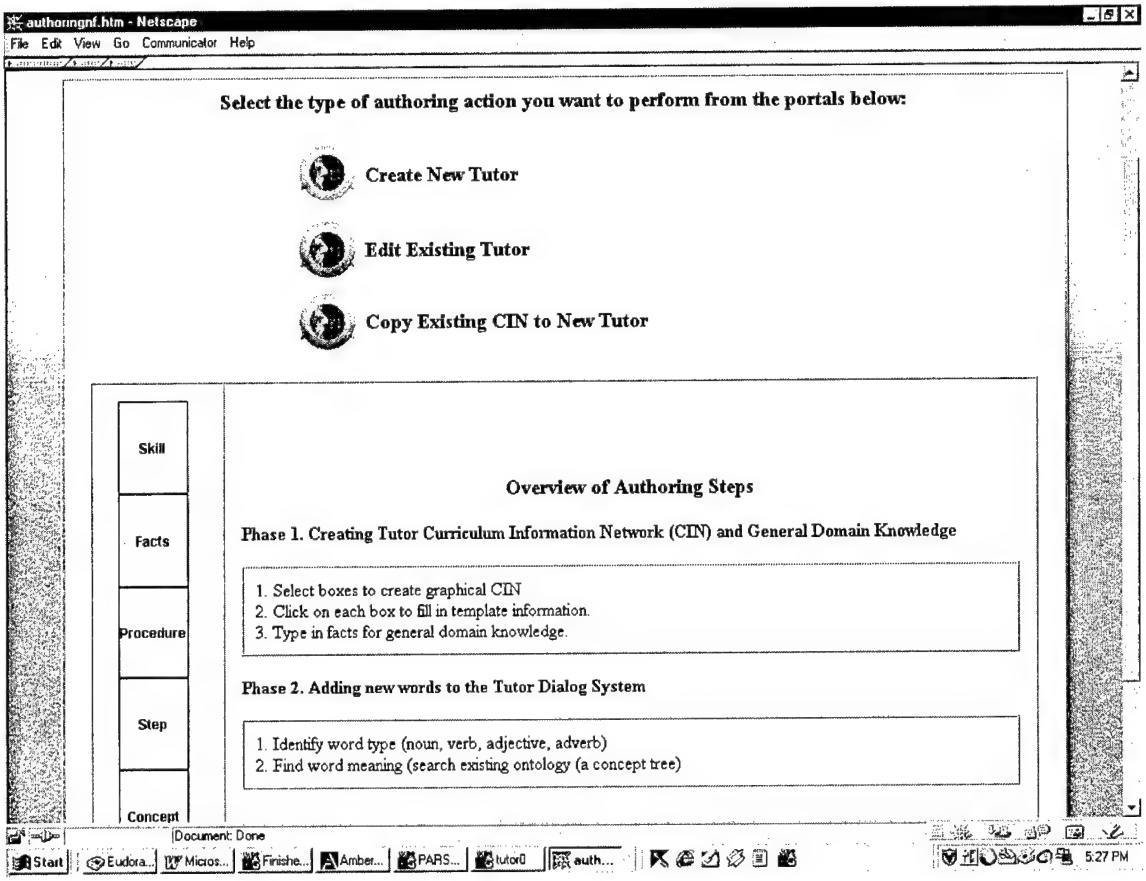
Initially a new character can only be a conversational agent. To change it to a tutorial agent we must add a CIN (curriculum information network). The CIN is a graph that represents the different kinds of skills to teach (concept, procedure, procedure step, facts, or skill for a composite or generic representation of the others). It also represents the relationships between the subject matter skills.

An instructional plan interpreter (the Tek Agent) interprets a pre-stored instructional strategy for presenting the curriculum graph, and for reviewing it if necessary. A user model is also built behind the scenes, and updated according to user answers to questions. The final report discusses the representation of instructional strategies and their interpretation in more detail.

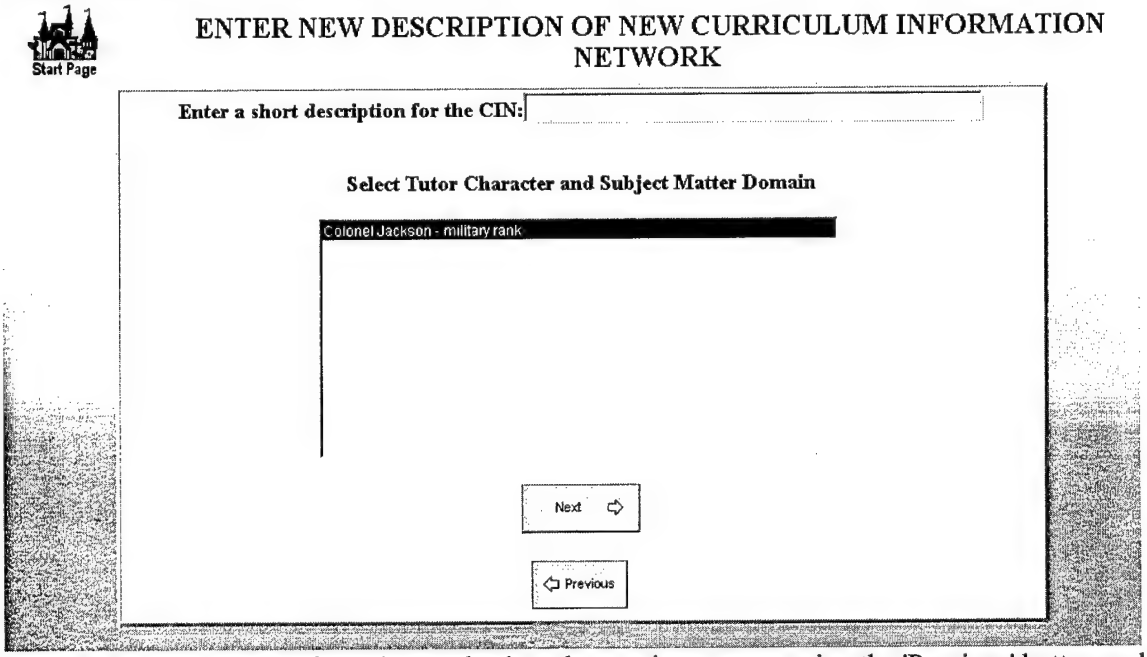
From the top-level screen select the authoring portal, i.e., the third option: Author (create or edit) a Tutor or Conversational Agent. On the next screen, the Authoring Portal, select the second option: Make it a Tutor Agent (Add Topics and Skills to Teach).



The screen after lets you select how to create a tutor agent. Select 'Create New Tutor' on the 'Authoring Tutor' screen.



The screen *after that* shows a list of just those conversational characters that do not have CINs. These are the characters that are not currently tutorial agents. Select one and also enter a short description for the tutorial material to be covered in this CIN.

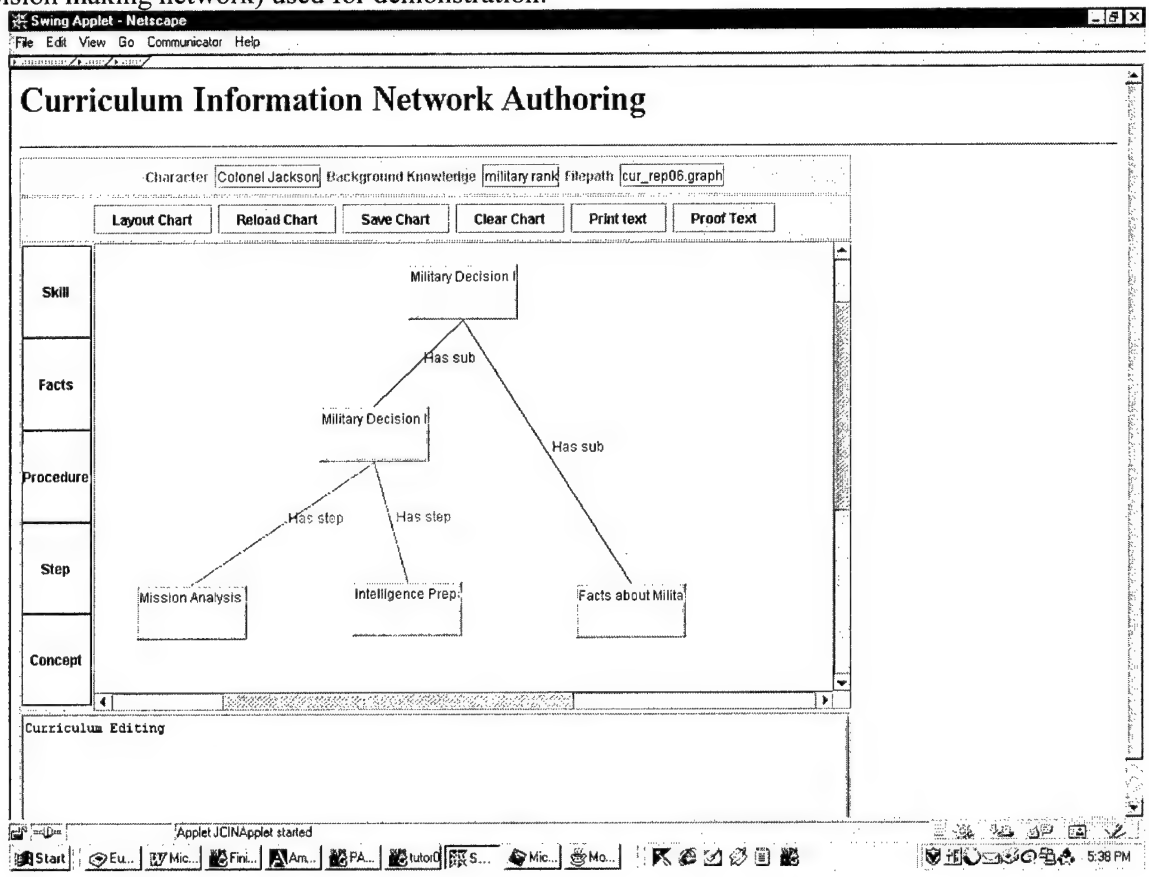


(If there is none to select from then go back to the previous screen using the 'Previous' button and then select 'Edit Existing Tutor'. If there are none there then no conversational characters exist and at least one

must be created to proceed). If all character slots are filled and you wish to start over with a completely blank slate, run c:\tutor\newstory.bat, but be careful to save your work as a scenario first!

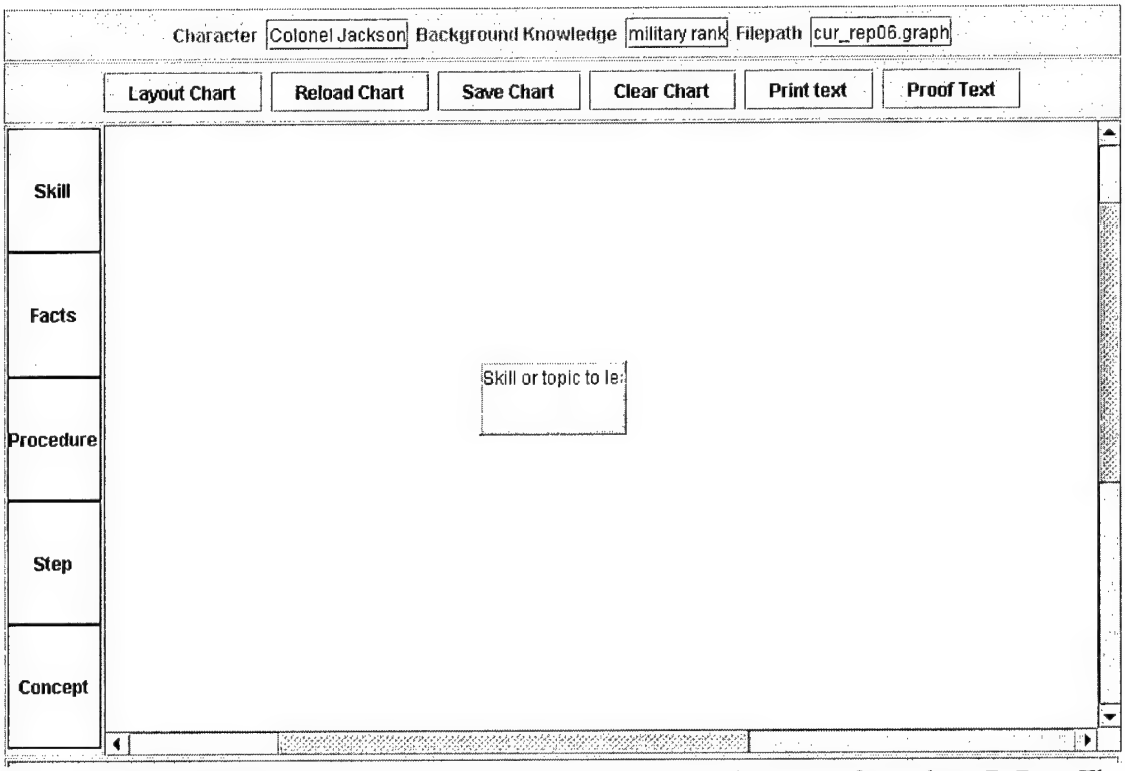
Creating a curriculum information network (CIN)

At first a default curriculum information network will be displayed, this is the 5-node MDMP (military decision making network) used for demonstration:



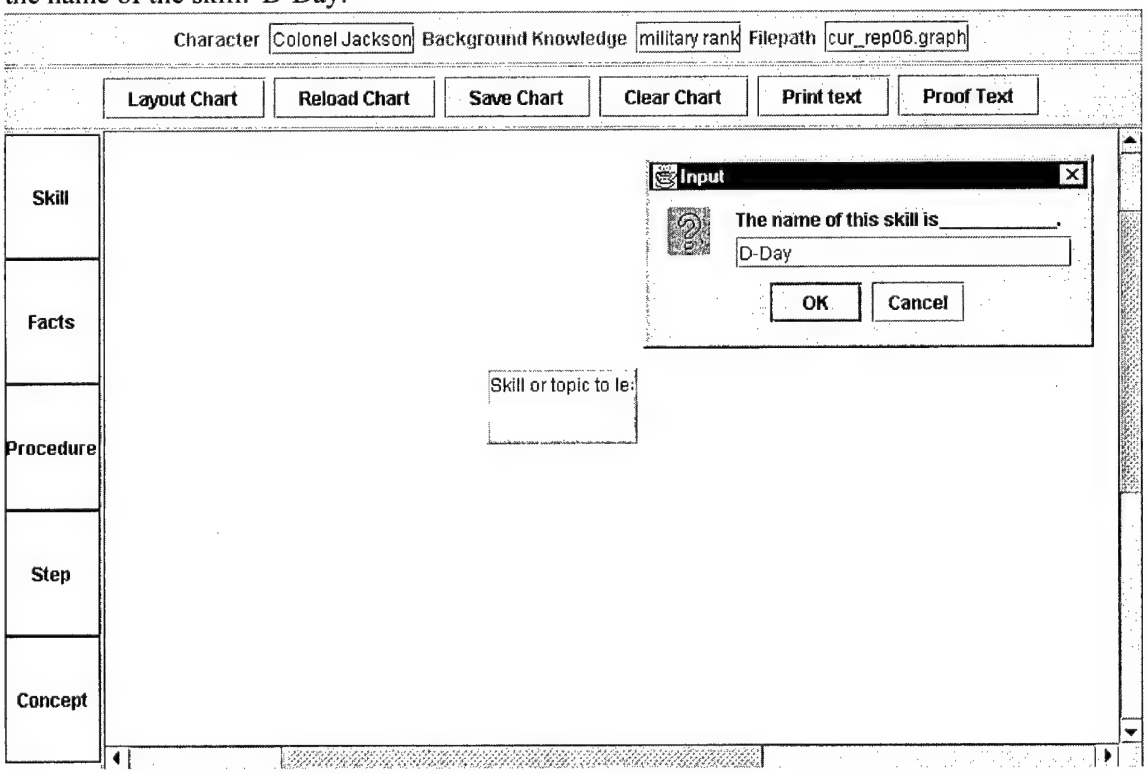
To clear this and start a new CIN from scratch select the 'Clear Chart' button. Note that you can drag the nodes around in the network to lay them out as you wish, although you may have to refresh the graph (minimize then restore the applet if necessary) in the process.

To edit nodes, right-click on a node then select 'Edit' from the pop-up menu. This will allow you to enter values for the node anew. For example, suppose we clear the current graph. Now just one node appears. The node represents the top-level skill to be taught.

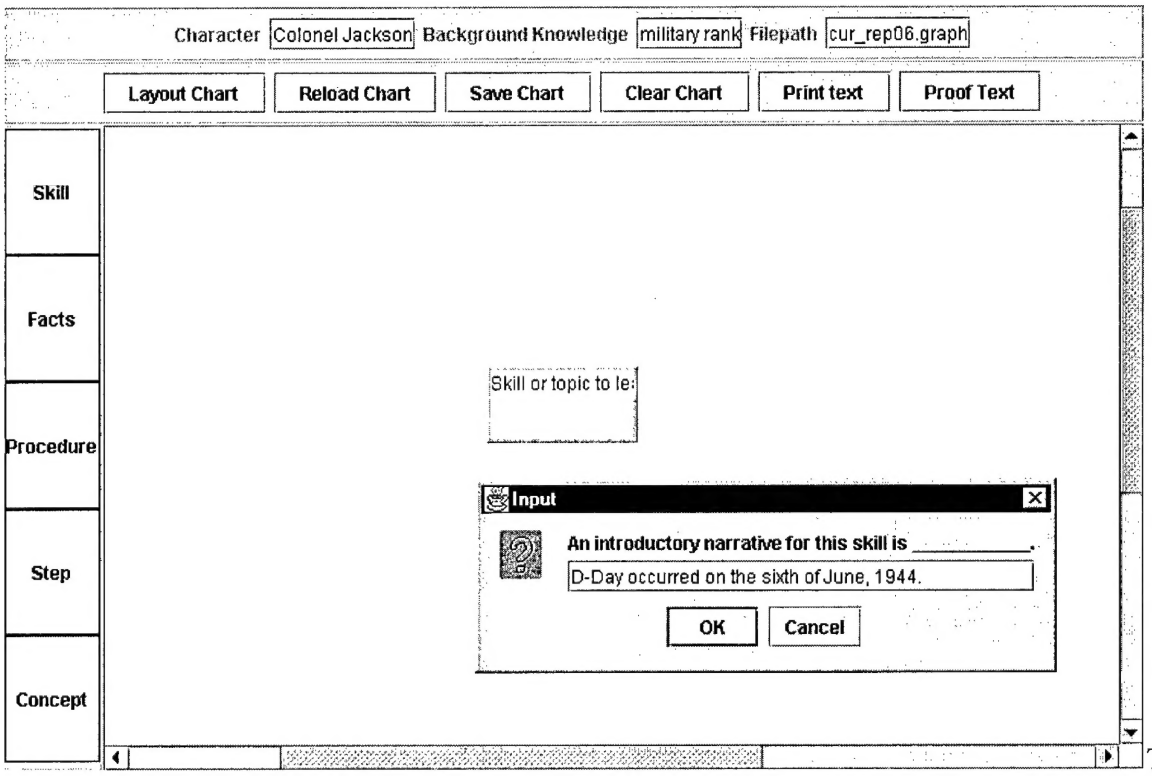


Suppose we wish to teach about D-Day. In this case the skills to be learnt are facts about D-Day. The top-level skill we will call 'D-Day'. The skills below this will be simple 'Facts' nodes as all that needs to be learnt are facts about D-Day.

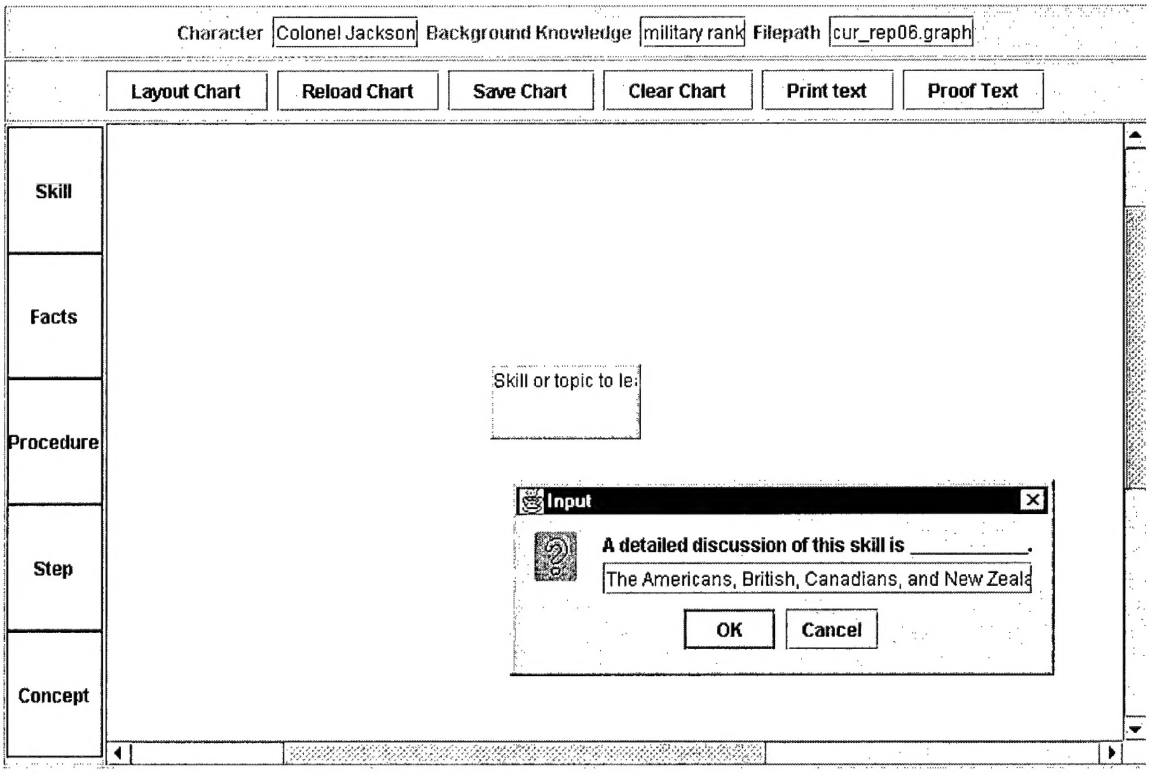
First we right-click on the Skills node. Then we enter the information requested by the dialog boxes. First the name of the skill: 'D-Day.'



Then a short introduction: 'D-Day occurrred on the sixth of June, 1944.'



Then a more detailed discussion: 'The Americans, British, Canadians, and New Zealand landed troops on the beaches of Normandy on D-Day.'



Note that several sentences could be entered on each of these. Next we are asked for a short review and we enter 'D-Day was the Allied attempt to retake Europe and crush the Nazi empire.'

How to edit a curriculum information network

To add new facts below this node drag a node of the proper kind from the left-hand palette of node-types. In this example we wish to add facts about the landings and the timing of D-Day.

We drag one Facts node to the D-Day node and drop it on that node. A new Facts node appears below it. We repeat this process and now have two Facts nodes below the main node. What this graph represents now is that to learn about D-Day the facts about both landing and the weather must be learnt. Of course the actual subject matter would be far, far more complex, but this is sufficient to illustrate how to use Tutor and edit CINs.

How to add learning objects

Edit one facts node, to give it a name 'Landings', an introduction 'The Normandy Landings occurred on five beaches, code-named Juno, Gold, Omaha, Sword, and Utah.' For detail, enter 'The Americans landed on Omaha and Utah beaches. The Canadians and British landed on Gold, Juno, and Sword.' For learning objects, say yes, and enter **http://normandy.eb.com/** for the text of the URL.

How to add question-answer pairs

For questions, enter one. The question is "What were the beaches that the Allies landed on code-named?" with expected answer "Omaha, Utah, Juno, Gold, and Sword."

Now edit the other facts node similarly. It's name is 'Timing' and has introduction 'The Allies needed to land at low tide during a full moon in weather that was not too rough for the sea transports.' For detail, use 'Eisehower had to postpone the initial D-Day landings 24 hours until a break in the weather allowed the landings to go forward.' For review, 'Weather played a key element in the timing of when the D-Day landings could occur.' No learning objects are entered. For Q/A the query is "When was D-Day originally planned for?" and the answer is "June 5".

Now select 'Proof Chart'. The chart will be saved and the relevant text extracted for proofing by the Amber agent and OLA Authoring system. The jump to OLA may take up to about four and a half minutes so be patient. The "Please wait..." screen refreshes three times before the jump completes.

Once in OLA each relevant text to be proofed will be presented. Note that the facts proofed are used in sidebar conversations so their format can be much simpler than the text presented in the narratives. In other words, much more complicated syntax can be used in presentations than might be acceptable for facts for the system to learn, e.g., we may present 'Three airborne divisions were dropped to secure the road to Arnhem: the British 1st Airborne in Arnhem, the American 101st (Screaming Eagles) near Nijmegen, and the American 82nd (Red Devils) near Eindhoven.' to the user while breaking this sentence up into 4 much simpler sentences for the parser and reasoning system to handle ('Three divisions were used in Operation Market Garden.', 'The American 101 st division was dropped near Nijmegen.', ...).

TROUBLE SHOOTING

Tutor is a complex set of programs that communicate by HTML, sockets, and DLLs. Sometimes one or more programs will get out of synchronization with the others or mistakes will occur if a shutdown is not clean (e.g., if the machine is rebooted while Tutor was running).

Hang or freeze with error message in parser window

Some things the parser cannot handle at first. For example, defining the terms for the 101st or 82nd Airborne Divisions can lead to problems in that Tutor does not at present handle terms like '101st which start off with numbers but that include characters. Acronyms like "D.C." could cause similar problems.

Netscape will not start

Try rebooting. Sometimes happens if a separate instance of Netscape was up when launching Tutor. Remember no other programs should run at the same time as Tutor.

Tutor0 shuts down immediately

Check to make sure there is not a file c:\filereaders\shutdown.txt. If there is delete the file. That should have been cleaned up automatically but sometimes gets left around after an improper shutdown.

Other problems

Text or curriculum information network graphics need refreshing

Try minimizing the window then restoring it. Also try not dragging the 'thumb' of the scroll bar but instead clicking above or below it.

Programs launched more than once (e.g., two copies of AmberServer running)

Do not accidentally double-click the launch shortcut more than once. This could lead to multiple copies, e.g., of AmberServer running. If you ever see more than one copy of AmberServer running then shut down all Tutor programs and relaunch it.

Background programs unrelated to Tutor interfere with Tutor resource requirements

Do not run other cycle-intensive programs or programs that may pop-up windows while Tutor is running. In general it is best to run Tutor entirely by itself.

Unrelated Windows operating system freeze

Sometimes Windows or Netscape will freeze unrelated to Tutor. In this case reboot the machine.

Learning object requires uninstalled application or plug-in

If the author stores a learning object that requires a plug-in that is not on the user's machine then the object cannot be displayed. E.g., if the learning object requires Adobe Acrobat to view then the user must have that application.

Hidden modal dialog box (beeps and will not continue until input entered)

If Tutor pops up a dialog box and you switch to another program,

FREQUENTLY ASKED QUESTIONS

How long should I wait...?

Here is some timing information taken off a 500 Mhz machine, 256 MB RAM, Pentium II, running Windows 98, and Netscape 4.5.

Time to launch from clicking Tutor.bat to appearance of top-level screen:
55 seconds.

Time to launch a conversational character after selecting the particular character:
2 minutes.

Time to launch a tutorial agent after selecting the agent and CIN:
1 minute 20 seconds to 1 minute 50 seconds

Time from selecting a CIN for editing to appearance in JCIN applet loaded completely:
1 minute

Time from requesting proof of a CIN to 'Chart Verified Saved':
15 seconds or less

Time from requesting proof of a CIN to 1st appearance of 'Please Wait...' screen:
1 minute

Time from requesting proof of a CIN to appearance of OLA authoring screen:
** 4 minutes 10 seconds to 4 minutes 15 seconds **

Your times will scale according to the speed of your machine. Note the last timing result: it will take about four and a half minutes from requesting that a CIN be proofed until the final jump to OLA.

What configuration is required...?

A fast machine, Pentium II or better, 500 Mhz or better, 256 MB or better RAM. Netscape, not Internet Explorer for the browser (tested in Netscape 4.5 and 4.7 primarily but appears to work in Netscape 6+ also). Java 1.2.2 (Java 1.3 appears to work OK, too, but if you have any problems with drag and drop in the JCIN applet, go back to Java 1.2.2.) Also do not display the Java control console while performing drag and drop operations, it appears to cause a problem.

What kind of sentences and queries can I use in authoring?

The final report will give a better idea. But use simple "You are..." kind of sentences for authoring facts for a particular character. Use "All x are y.", "All x have y.", and similar rules for authoring rules.
Examples:
You are a general.
All generals have stars on their uniforms.